TAMPEREEN TEKNILLINEN YLIOPISTO

LASSE YLINEN

TIETOPALVELUN PÄIVITTÄMINEN JULKAISIJA–TILAAJA-MALLIN MUKAISEKSI DATAPALVELIMEKSI

Diplomityö

Tarkastaja: professori Antti Valmari
Tarkastaja ja aihe hyväksytty
Tieto- ja sähkötekniikan
tiedekuntaneuvoston
kokouksessa 9. joulukuuta 2009

TIIVISTELMÄ

TAMPEREEN TEKNILLINEN YLIOPISTO
Tietotekniikan koulutusohjelma
YLINEN, LASSE: Tietopalvelun päivittäminen julkaisija–tilaaja-mallin mukaiseksi datapalvelimeksi
Diplomityö, 81 sivua, 1 liitesivu
Toukokuu 2010
Pääaine: Ohjelmistotuotanto
Tarkastaja: professori Antti Valmari
Rahoittaja: Patria
Avainsanat: tietopalvelu, datapalvelin, hajautettu järjestelmä, vertaisverkko, julkaisija–tilaaja-malli, MANET

Tämän työn aiheena oleva datapalvelin tulee olemaan keskeinen osa yrityksen PSVSP-ohjelmistoperheen uudistettua ydintä. Datapalvelimen tehtävänä on mahdollistaa viestien tietoturvallinen välittäminen siihen liittyneille asiakasmoduuleille epäluotettavassa ja turvattomassa verkossa, sekä erottaa nämä asiakkaat toisistaan. Asiakkaat voivat sijaita fyysisesti pitkienkin matkojen päässä toisistaan ja olla erillisissä verkoissa kytkeytyneinä.

Tehtävän suorittamiseksi datapalvelin kannattaa toteuttaa hajautettuna järjestelmänä. Lisäetuja saavutetaan hyödyntämällä tilaaja–julkaisija-mallin etuja sekä toteuttamalla vertaisverkkomainen rakenne datapalvelimen eri yksiköiden välille. Näin datapalvelimen asiakkaiden ei tarvitse tietää toisistaan, vaikka niiden välillä kulkisikin viestejä.

Salausten käyttäminen tietoturvan parantamiseen voi aiheuttaa ongelmia yritettäessä hyödyntää sekä hajautusta että tilaaja–julkaisija-mallia. Jos kuitenkin käytetään epäsymmetrisiä salausmenetelmiä ja salataan hajautetun verkon eri yksiköiden välinen liikenne jokaisen välin yli erikseen, voidaan molempia hyödyntää yhdessä.

Datapalvelimen tulee lisäksi mahdollistaa viestien rakenteen muuttuminen tarpeen mukaan. Tämä tapahtuu erillisen tulkki-moduulin toimesta, joka voi vaihtaa tiedon muotoa vanhempaan rakenteeseen. Tämän erillisen moduulin avulla jonkin tiedon tarvitsijaa ei tarvitse muuttaa, mikäli kyseisen tiedon tuottaja muuttaa sen muotoa. Tämä johtaa parempaan ylläpidettävyyteen, kun pienet muutokset eivät vaadi muutoksia useisiin eri paikkoihin.

ABSTRACT

TAMPERE UNIVERSITY OF TECHNOLOGY
Master's Degree Programme in Information Technology
YLINEN, LASSE: Upgrading data service to a data-server with publisher-consumer model
Master of Science Thesis, 81 pages, 1 Appendix page
May 2010
Major: Software engineering
Examiner: Professor Antti Valmari
Funder: Patria
Keywords: data service, data server, distributed system, peer-to-peer network, publisher-consumer model, MANET

The data server, which is the topic of this thesis, will be a central part of a PSVSP software suite that the company offers. The purpose of the data server is to allow safe transmitting of messages and information to customers in an untrustworthy and weak quality network, and to separate these customers from each other. These customers may have long physical distance in between them and they may be connected to separate networks.

In order to succeed in its task, the data server should be a distributed program. Additional advantages can be gained by utilizing both the publisher-consumer model and peer-to-peer networking. In this way, one customer need not know the other, even if they are sending messages to each other.

Utilizing encryption for enhancing safety of information can pose problems when utilizing distributed programming in the publisher-consumer concept. However, by using asymmetric encryption methods to encrypt on a hop-by-hop basis these methods can still be used together.

The data server has to also allow changes to the structure of the information it transmits. A separate Translator module is used for this purpose, and it can transform information from a newer to an older format. With the help of this module, the customer which uses the information need not be changed when the producer of this information changes the format of information to a different structure. This leads to better maintainability, as small changes to the layout of the information need not be replicated to multiple locations or programs.

ALKUSANAT

Tämä on Tampereen teknillisen yliopiston Tietotekniikan koulutusohjelman Ohjelmistotuotannon pääaineeseen tehty diplomityö. Työn aiheena on PSVSP-ohjelmistoperheen tuoterunkoon kuuluvan tietopalvelun päivittäminen julkaisija–tilaaja-mallin ja vertaisverkon periaatteiden mukaisesti toimivaksi datapalvelimeksi, tähän liittyvien tarpeiden tunnistus sekä ohjelman suunnittelu näiden periaatteiden mukaisesti. Varsinainen toteutus on rajattu työn ulkopuolelle.

Työni teossa olen saanut apua lukuisilta tahoilta. Haluan erityisesti kiittää

- ohjaajaani Tero Kaipiota suuntien ja ongelmien osoittamisesta
- Patriaa erittäin mielenkiintoisesta oppi- ja työmahdollisuudesta
- vaimoani Sarah Ylistä kärsivällisyydestä ja kannustuksesta
- työkavereitani eri ratkaisuvaihtoehtojen ja aiheen laajasta käsittelystä sekä esilukijoina toimimisesta sekä
- tarkastajaani professori Antti Valmaria tarkasta ohjeistuksesta

SISÄLLYS

SISÄLLYS (KUVAT, LISTAUKSET JA TAULUKOT)

TERMIT JA NIIDEN MÄÄRITELMÄT

AAA-malli	Authentication, Authorization and Accounting model on yleisesti käytössä oleva tapa toimia järjestelmän tietoturvallisuuden edistämiseksi.
ACE-KEM	Advanced Cryptographic Engine Key Encapsulation Mechanism on tapa kätkeä tiedon salausavain.
ACE-sovelluskehys	Adaptive Communication Environment framework on sovelluskehys verkon käyttämiseksi sellaisissa tapauksissa, joissa tarvitaan käyttöympäristön ja kehityskielen riippumattomuutta sekä hajautuksen tukemista.
AES	Advanced Encryption Standard on turvalliseksi luokiteltu tietojen salausmetodi, jonka perässä toisinaan ilmoitetaan salaukseen käytetty bittimäärä (esimerkiksi AES-256).
API	Application Programming Interface on ohjelman käyttämä rajapinta jonkin toisen ohjelman tai sen osan toteuttamien palveluiden hyödyntämiseksi.
Asiakasmoduuli	Moduuli, joka toimii PSVSP-tuoterungon päällä ja hyödyntää datapalvelimen tarjoamia viestinvälityksen palveluita
Broadcast	Katso kohta yleislähetys.
CBIS	Content Based Information Security tarkoittaa viestin sisältöön pohjautuvaa tietoturvallisuutta.
Certificate	Katso kohta varmenne.
Datapalvelimen yksikkö	Yksittäinen datapalvelimen toimintaan osallistuva yksikkö, joiden kokonaisuus muodostaa datapalvelimen.

Datapalvelin	Uusi, palvelimen kaltaisena toimiva tietoa välittävä palvelu, joka muodostaa kokonaisuuden useista yksiköistä.
DOD-malli	Yhdysvaltojen Department of Defence:n luoma tapa jakaa järjestelmien toimintoja hoitavat osat eri kerroksiin, jotta ne voidaan liittää toisiinsa helpommin. Katso myös OSI-malli.
J&T-rekisteri	Rekisteri, jossa pidetään yllä julkaisujen ja tilausten tietoja.
MANET	Mobile Ad hoc NETwork on erilaisten liikkuvien päätelaitteiden itsenäisesti muodostama verkko.
MITM	“Man In The Middle” hyökkäyksessä vihamielinen taho asettuu kahden keskustelua aloittavan osapuolen väliin ja huijaa osapuolet luulemaan, että ne keskustelevat suoraan toistensa kanssa keskustelun välittyessä vihamielisen tahon kautta.
OSI-malli	Open Systems Interconnection model on tapa kuvata järjestelmän osat eri kerroksiin niin, että yhteenliittyminen muihin osiin on helppo toteuttaa. Tukee kerroksen vaihtamista toiseen muita kerroksia muuttamatta.
PSEC-KEM	Provably Secure Elliptic Curve Key Encapsulation Method on tapa kätkeä salausavain elliptiseen käyrään pohjautuvalla menetelmällä.
PSVSP	Nimitys yrityksen omalle sovelluskehykselle, jonka toimintaan datapalvelin liittyy. Katso myös kappale 3.3.
REGEX	Regular expression eli säännöllinen lauseke antaa mahdollisuuden hakea ja tarkastaa tekstin säännönmukaisuuksia.
Salausavain	Tietojen salaamiseen tai purkamiseen käytetty sarja merkkejä, joka toimii avaimena salatun ja salaamattoman tiedon välillä.
SHA	Secure Hashing Algorithm on tapa luoda varmistuskoodi tietojen sisällöstä.

Sovelluskehys	Ohjelmiston yleiskäyttöinen osa, jonka päälle rakennetaan varsinainen erikoistettu ohjelma. Yleensä sovelluskehyksen ohjelmakoodia ei voida muokata, jolloin muokkaustarpeen yhteydessä joudutaan usein toteuttamaan erikoistettu versio sovelluskehyksen tarjoamasta palvelusta – yleensä tällainen on vähintään ohjelman käynnistämiseen liittyvien toimenpiteiden ohjaus. Sovelluskehys pitää ohjelman suorituksen kontrollin itsellään, mutta kutsuu näitä erikoistettuja versioita palveluidensa toteutuksista.
Tarkastussumma	Merkkijono, joka on laskettu muista tiedoista ja jonka avulla yleensä pyritään toteamaan mahdolliset muutokset tiedossa.
Tietopalvelu	Edeltävä versio tietoja välittävästä palvelusta, jonka päivitystä tämä dokumentti kuvaa.
Varmenne (Certificate)	Varmenteella todistetaan esimerkiksi julkisen avaimen olevan joltakin tietyltä taholta ympäristöissä, joissa ei joko ole luotettua tahoa tai varmaa yhteyttä siihen. Voi sisältää myös muita tietoja kuin pelkän julkisen avaimen ja allekirjoituksen.
XML	eXtensible Markup Language on tapa kuvata puumaisia tietoja tekstimuodossa.
Yleislähetys (Broadcast)	Yleislähetyksellä tarkoitetaan sellaista verkossa tapahtuvaa viestintää, joka lähetetään jokaiselle kuulijalle.

1. JOHDANTO

Tutkimusongelmana tämän työn aiheen voisi kuvata kysymyksellä ”Miten tällä hetkellä käytössä olevaa tietopalvelua voidaan kehittää tulevaisuuden tarpeet huomioiden?”. Tarkoituksena on paitsi vastata tähän kysymykseen, myös kuvata nykytilanne ja perustella tehdyt valinnat sen ja tulevaisuuden tarpeiden huomioinnin pohjalta.

Aluksi luvussa 2 esitellään nykyisen tietopalvelun toiminta ja visio sen kehittämisestä datapalvelimeksi. Tämän jälkeen luvussa 3 kerrotaan mitä datapalvelimella tarkoitetaan (eli millaisia tehtäviä sille on asetettu) ja millaisessa toimintaympäristössä se toimii. Kun nämä on käsitelty, kerrotaan luvussa 4 datapalvelimen tietoturvasta sekä luvussa 5 minkälaisia tietoja sillä välitetään. Luvussa 6 käydään läpi datapalvelimen rakentamiseen tarvittavat osat ja mitkä ovat kunkin osan tehtävät, sekä sen toimintatavat asiakasmoduulien suhteen luvussa 7 ja vastaavasti toisten datapalvelimen yksiköiden suhteen luvussa 8. Luvussa 9 suoritetaan yhteenveto.

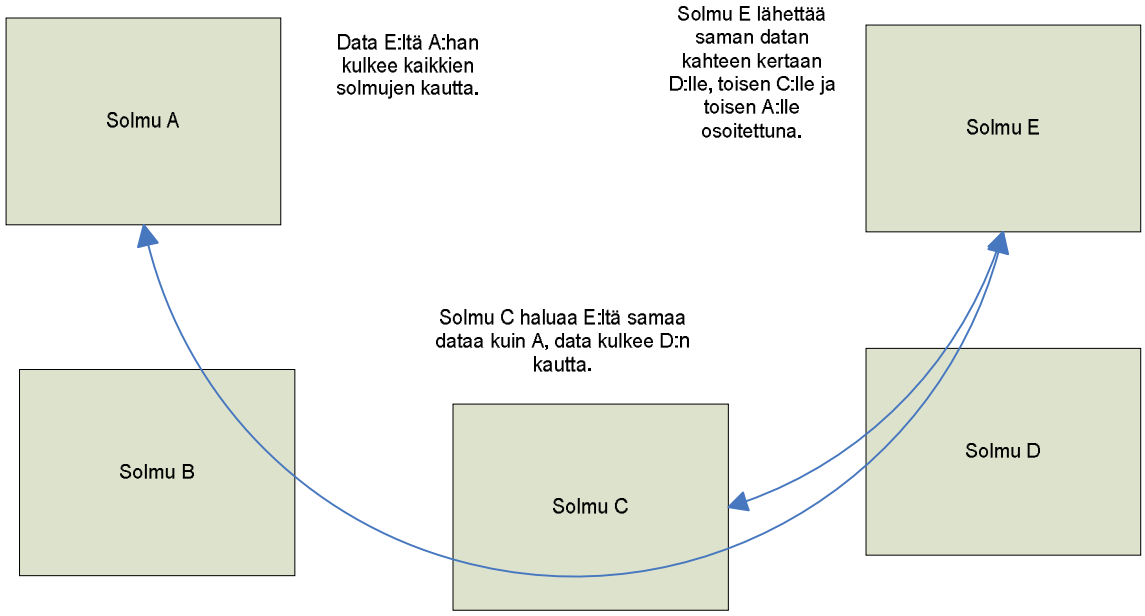
Dokumentissa käytetään runsaasti Internetistä saatavilla olevia lähteitä. Syynä tähän on tarve käyttää viimeisintä tietoa aiheesta. Pääosa käytetyistä lähteistä löytyy sieltä vapaasti saatavilla olevassa muodossa. Käytetyt tietolähteet on analysoitu riittävän luotettaviksi tutkimalla ja vertaamalla niitä tarjoavia tahoja. Työssä pyritään selventämään tärkeimpiä asioita esimerkkien avulla. Erityisesti lukujen 7 ja 8 esimerkkien avulla on pyritty selventämään kokonaisuutta.

2. TIETOPALVELUSTA DATAPALVELIMEEN

Tässä luvussa kerrotaan nykyiseen tietopalveluun tehtävistä muutoksista, joiden avulla siitä luodaan uusi datapalvelin-niminen kokonaisuus. Pohjaksi uuden datapalvelimen suunnitelmiin on valittu AAA- ja julkaisija–tilaaja-mallit. Nämä suunnitellun datapalvelimen pohjarakenteet esitellään omissa kohdissaan. Lopuksi tässä luvussa käydään läpi visio datapalvelimesta.

2.1. Miten nykyinen tietopalvelu toimii

Nykyisin käytössä olevan ohjelmistorungon version sisältämä tietopalvelu on varsin yksinkertainen, eikä se esimerkiksi tarkasta lähetysten suhdetta toisiinsa. Näin eri lähetyksissä voi kulkea samaa informaatiota osittain samaa reittiä, kun lopullinen kohde vaihtuu. Turhan liikenteen vähentäminen verkossa tapahtuu reitityksen tasolla. Ongelmia tulee esimerkiksi kuvan osoittamassa viiden solmun tilanteessa (kuva 2.1), jossa kaksi solmua (A ja C) haluavat samoja tietoja solmulta E. Vaikka yhteydet kulkisivat samaa reittiä, ei reitittäjän tasolla voi aina tunnistaa solmulle D menevän samaa dataa kahteen kertaan – esimerkiksi salauksen vuoksi. Datapalvelimen tasolla tämä kuitenkin näkyisi, ja siihen voitaisiin reagoida.



Kuva 2.1: Nykyisessä tietopalvelussa esiintyvät tietovirrat

Vaikka esimerkkikuva yksinkertaistaa nykyisen tietopalvelun toimintaa huomattavasti, tuo se esille verkossa olevan tehokkuusongelman: solmujen väleillä kuljetetaan moninkertaisesti dataa verrattuna optimaaliseen tilanteeseen. Koska verkkoon liittyvien solmujen ja siinä liikkuvien erilaisten datavirtojen määrän odotetaan kasvavan entisestään, halutaan toimintaa tehostaa datapalvelimen käyttöön ottamisen avulla. Näin voidaan ennakoivasti parantaa verkon toimintakykyä vähentämällä verkossa tapahtuvaa viestintää. Ohjelmistoperheen päivityksen yhteydessä voidaan samalla uudistaa verkon rakennetta ja toimintamalleja myös muilla tavoin.

Tilannetta voisi optimoida esimerkiksi salaamalla tieto C:lle asti, ja salaamalla tiedot uudestaan solmulta C kohti solmua A lähetettäessä – tämä täyttää lähes kaikkien luottamuksellisuusasteiden vaatimukset tiedoille. Poikkeuksen muodostavat voimakasta salausta vaativat tiedot, koska niiden pitää olla salattuina aina, kun niitä ei suoraan käsitellä [24, s. 64 kohta 10]. On myös mahdollista, ettei tietoa välittävällä datapalvelimen yksiköllä ole avainta tai oikeuksia tiedon purkamiseen. Myös tällaisessa tilanteessa tietovirtojen yhdistäminen on mahdotonta, koska tiedon rakennetta ei tällöin tiedetä eikä siten voida havaita samasta tiedosta löytyviä osatietoja.

2.2. Hyödynnettävät suunnitteluperiaatteet ja rajapintamalli

Tässä kohdassa kerrotaan aluksi AAA-mallista ja sen hyödyntämisestä tietoturvan parantamiseen. Tästä edetään kertomaan julkaisija–tilaaja-mallin hyödyntämisestä tietovirtojen hallinnan menetelmänä. Lopuksi kerrotaan rajapinnasta datapalvelimeen.

2.2.1. AAA-malli lyhyesti

Kuva 2.2 esittää AAA-mallin [18] mukaisen toimintaketjun. Mallin nimen jokainen A-kirjain vastaa yhtä toimintojen osa-aluetta. Englanniksi nämä edustavat seuraavia toimia: Authentication, Authorization ja Accounting. Suomeksi ne voidaan kääntää esimerkiksi seuraavasti:

- autentikointi eli vastapuolen luotettava tunnistaminen
- auktorointi eli varmistetun osapuolen oikeuksien kontrollointi, sekä
- tapahtumien kirjaus, joka käytännössä tarkoittaa lokitietojen keräämistä käyttäjien toimista.

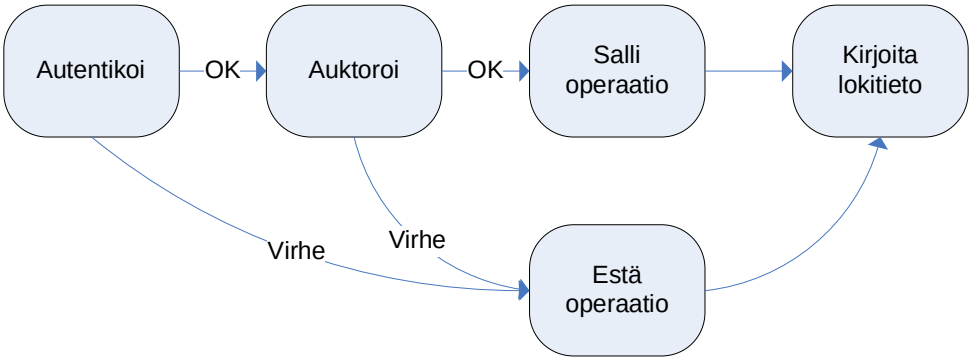
Koska PSVSP-ohjelmistoperhe on turvallisuuskriittinen järjestelmä, kaikkien sen osien tulee mahdollistaa tämän periaatteen mukaan toimiminen. Mallia noudattava datapalvelin varmistaa ensimmäisenä asiakkaaksi pyrkivän tahon ”henkilöllisyyden” (hyväksytäänkö se asiakkaaksi). Tämän jälkeen tarkastetaan onko kyseisellä asiakkaalla oikeutta operaatioon, jonka se halusi suoritettavan, ja voidaanko sitä suorittaa käytettävissä olevilla työkaluilla. Lopuksi datapalvelin kirjaa nämä pyynnöt ja niiden suhteen tekemänsä päätökset lokitietoihin.

Autentikointiin auttaisi erityisesti luotettavan kolmannen tahon oleminen verkossa. Tätä ei kuitenkaan voida olettaa mahdolliseksi, koska datapalvelimen toimintakyky on pystyttävä säilyttämään myös verkon osittuessa erillisiin lohkoihin – jokin tällainen

osittuminen voi eristää jonkin osan verkkoa luotettavasta tahosta. Eräs ratkaisu olisi tuntea kaikkien muiden datapalvelimen yksiköiden henkilöllisyyden varmentaviin varmenteisiin sidotut julkiset avaimet.

Tällaisessa tilanteessa henkilöllisyys voitaisiin varmistaa kohtuullisen yksinkertaisesti esimerkiksi salaamalla jokin haaste etukäteen tiedossa olleella julkisella avaimella. Myös toisen osapuolen tulee tällöin varmistua osapuolen aitoudesta, joten se lähettäisi takaisin vastauksen lisäksi oman haasteensa alkuperäisen haasteen lähettäneen tahon julkisella avaimella salattuna viestinä. Haasteena voi toimia esimerkiksi kahden ison satunnaisluvun yhteenlasku – ilman kohteen yksityistä avainta on huomattavan vaikeaa arvata vastapuolen valitsemat kaksi lukua (eli mikä on oikea summa), mutta mikäli vastapuoli on se kuka väittää olevansa, saa se viestin purkamisen kautta luvut haltuunsa ja kykenee suoriutumaan tehtävästä helposti.

Näitä haasteita voidaan myös tehdä useita varmuuden kasvattamiseksi, jos vaaditaan tavanomaista suurempaa turvallisuutta. Vaihtoehtoisesti voidaan salata vastaus toisen osapuolen julkisen avaimen avulla, jolloin vastauksen oikeellisuudesta voi tehdä päätelmiä vain oikea taho – tosin vihamielinen taho voi ohittaa tämän oikeellisuuden tarkastuksen ja käynnistää hyökkäyksiä verkkoa vastaan, jos toinen osapuoli ei esitä omaa haastettaan vaan odottaa joko yhteyden hylkäystä tai hyväksymistä. Näitä hyökkäysmahdollisuuksia on esimerkiksi tietoverkon kapasiteetin täyttö ja virheellisen tiedon syöttäminen järjestelmään, sillä kummassakaan ei yleensä ole välttämätöntä saada käsiin vastaanottajan lähettämiä viestejä, jolloin vastauksien salattuna oleminen ei ole este. Tämä avainten hallintaa vastaava tehtävä ei kuitenkaan ole datapalvelimen vastuulla.



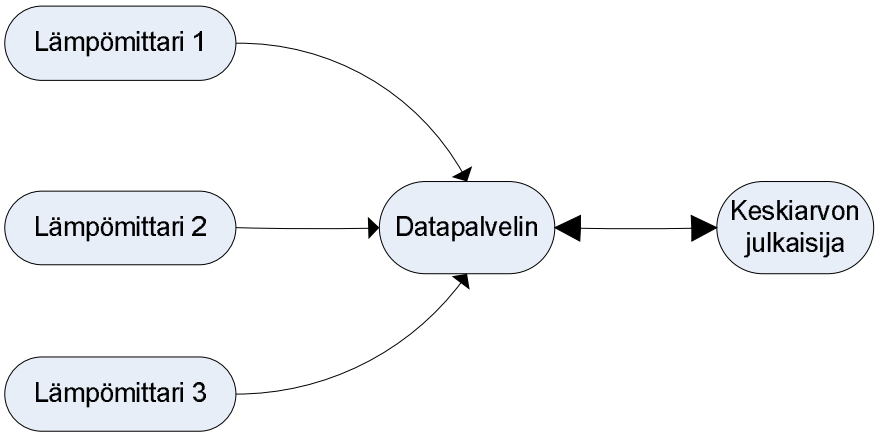
Kuva 2.2: AAA-mallin mukainen toimintaketju

Kaikki päätökset kirjataan lokiin. Näiden lokitietojen perusteella voidaan reagoida erilaisiin realisoituneisiin turvallisuusuhkiin analysoimalla ne sekä tunnistamalla siellä olevista jäljistä mahdollisia hyökkäyksiä verkkoa vastaan. Tämä tarkastelu ja reagointi on kuitenkin rajattu tämän työn ulkopuolelle, ja sen oletetaan tapahtuvan ulkopuolisen tahon tekemänä. Valtiovarainministeriön vaatimuksen mukaan salassa pidettäviin tietoihin koskijoista pitää aina jäädä jälki [25, sivu 68], ja lokien ylläpitäminen riittää täyttämään tämän vaatimuksen.

2.2.2. Julkaisijoista ja tilaajista

Julkaisija–tilaaja-mallissa tiedolla on julkaisijoita ja tilaajia. Näistä julkaisijat tuottavat tietoa ja tilaajat kuluttavat sitä. Mallin esittely ja analysointi sen eroista joihinkin vastaaviin tapoihin toimia löytyy esimerkiksi teoksesta [5]. Yksittäinen toimija voi olla myös molemmissa näistä rooleista, esimerkiksi ison alueen keskimääräistä lämpötilaa julkaiseva toimija voi tilata alueen paikkakuntien lämpötiloja tuottaakseen itse julkaisemaansa tietoa. Tästä on muodostettu graafinen esimerkki (kuva 2.3), jossa nuolet datapalvelimelle kuvaavat tiedon julkaisua ja nuoli asiakasmoduuliin kuvaa vastaavasti tilausta.

Kun malliin tehdään datapalvelimen kaltainen välikäsi hoitamaan tilaukset, julkaisut ja jakelun, ei julkaisevan moduulin tarvitse tietää vastaanottavasta (tai vastaanottavista) moduulista mitään muuta kuin niihin liittyvien oikeuksien tarkistamisen vaatimat osat. Vastaavasti tilaajan ei tarvitse tietää sitä, mitä kautta tai keneltä tilattu tieto saapuu – samoin kuin esimerkiksi kirjeen vastaanottaja ei yleensä välitä onko sen toimittanut DHL vai Itella ja minkä konttoreiden kautta se on kulkenut. Datapalvelimen ja asiakasmoduulin näkökulmat toiminnasta eroavat toisistaan. Tätä eroa toiminnan kuvasta on selvennetty graafisesti (kuva 2.4).

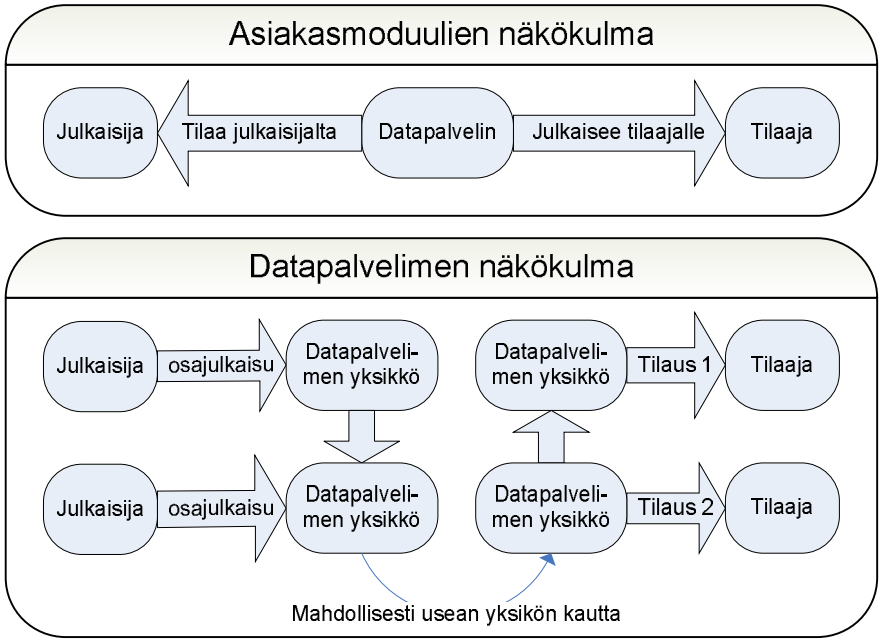


Kuva 2.3: Toimiminen sekä julkaisijan että tilaajan roolissa

Yksittäisen tilaaja- tai julkaisijamoduulin tulisi olla yhteydessä aina vain tasan yhteen datapalvelimeen. Koska datapalvelimen yksiköt välittävät toisilleen ainoastaan tilattua tietoa ja nämäkin tilaukset pyritään ketjuttamaan tehokkaasti, ei verkossa tapahdu ylimääräistä tiedonsiirtoa. Myös salatun tiedon usealle kohteelle välittämisen yhteydessä datapalvelin kykenee estämään usean tietovirran syntymisen – tosin tähän aiheuttaa poikkeuksen tilanne, jossa datapalvelimella ei ole oikeutta käsitellä tietoa salaamattomana. Näin verkon siirtokapasiteettia voidaan käyttää huomattavasti optimaalisemmin, jolloin verkon yli saadaan myös siirrettyä suurempia määriä tietoja kuin mitä saataisiin siirrettyä tehtäessä tietojen lähetyksen yhdistäminen reitityksen tasolla.

Tarve tietojen salaamiseen aiheuttaa kuitenkin ongelman mallin soveltamisessa. Jos halutaan lähettää sama viesti kahdelle eri taholle, joudutaan puhtaissa julkisen salauksen menetelmissä käyttämään kahta eri salausavainta. Tällöin viestin kohteille tulisi varsin

erilaiset viestit, vaikka niiden sisältö olisikin samaa. Tämä voidaan kuitenkin ratkaista käyttämällä kohdassa 4.2.3 esiteltävää hybridimenetelmää, eli salaamalla tieto tietokohtaisesti yksityisellä avaimella ja lähettämällä tämä avain erikseen jokaiselle vastaanottajalle julkisen salauksen menetelmin. Toinen salauksen aiheuttama ongelma on tiedon kulkureitin välillä olevien yksiköiden jääminen ilman avainta, jolloin ne eivät voi hyödyntää lävitseen kulkemia osatietoja.



Kuva 2.4: Julkaisut ja tilaukset asiakasmoduulin ja datapalvelimen kannalta

Näin sama tieto kulkisi verkossa kahteen kertaan, eikä mallin etuja päästä hyödyntämään. Datapalvelimen yhteydessä tämä ongelma on ratkaistu pyrkimällä käyttämään datapalvelimen yksiköiden välisessä liikenteessä kunkin yksikön omaa julkista avainta. Näin yksikkö kykenee purkamaan viestin ja tämän jälkeen salaamaan sen vastaanottajien omia avaimia käyttäen. Tämä mahdollistaa tilausten ketjuttamisen eri datapalvelimen yksiköiden välillä, eikä se tule näkymään asiakkaille – niille datapalvelin on vain tiedon tilaaja tai sen tuottaja, todellisten tilaavien tai julkaisevien tahojen ollessa sen takana piilossa.

2.2.3. Ohjelmointirajapinta

Datapalvelin toteutetaan ohjelmointikielestä ja käytettävästä kääntäjästä riippumattomasti API-tekniikalla eli luomalla monikielinen rajapinta mahdollistamaan ohjelmiston käyttö muissa ohjelmistoissa. Tämän rajapinnan avulla asiakasmoduulit kykenevät liittymään johonkin datapalvelimen yksiköistä, ja niille luodaan puskurit sisään ja ulos tulevia lähetyksiä varten. Loput kommunikoinnista voidaan hoitaa puskureiden välityksellä.

Tämä rajapinta luodaan käyttämällä Adaptive Communication Environment-sovelluskehystä (ACE-kehys) [12]. Näin datapalvelimen toteutuskieli ei aseta

vaatimuksia tätä ohjelmiston osaa mahdollisesti tulevaisuudessa käyttävien asiakasmoduulien toteutukseen, vaan ne voidaan toteuttaa niille parhaiten soveltuvalla ohjelmointikielellä ja kääntäjällä. ACE-kehys tarjoaa lisäetuna PSVSP-ohjelmistoperheeseen myös käyttöjärjestelmäriippumattomuuden.

ACE-kehys on toteutettu C++-kielellä, mutta PSVSP:n yhteydessä siihen liitytään C-ohjelmointikielisen rajapinnan kautta. Tätä rajapintaa voidaan käyttää useiden eri ohjelmointikielten avulla kirjoittamalla rajapinnan käyttöön ylimääräinen rajapinta tuolla kielellä. Esimerkiksi Javalla tehtyyn ohjelmaan tehdään ensin rajapinta, joka käyttää ACE-kehyksen rajapintaa. Tämän jälkeen varsinainen tuotettava ohjelmistokomponentti käyttää Java-rajapintaa, joka vuorostaan käyttää ACE-kehystä.

2.3. Visio uudesta datapalvelimesta

PSVSP:n uuden version kehittämisessä halutaan parantaa myös verkon toimintaa. Ohjelmistoperheeseen kehitellään tulevaisuuden tarpeita varten uusia ominaisuuksia sisältävä reitityskerros sekä tehokkaampi ja edeltävän version tietopalvelua turvallisempi datapalvelin. Nämä osat ovat osana ydintä, johon myöhemmät ohjelmistoprojektit tulevat pohjautumaan. Näistä ytimen osista tässä työssä käsitellään lähinnä datapalvelinta, sillä reitityskerroksen päivitys on liitoksissa tästä työstä erilliseen projektiin. Aiheesta on kerrottu lisää Antti Viidanojan diplomityössä [26].

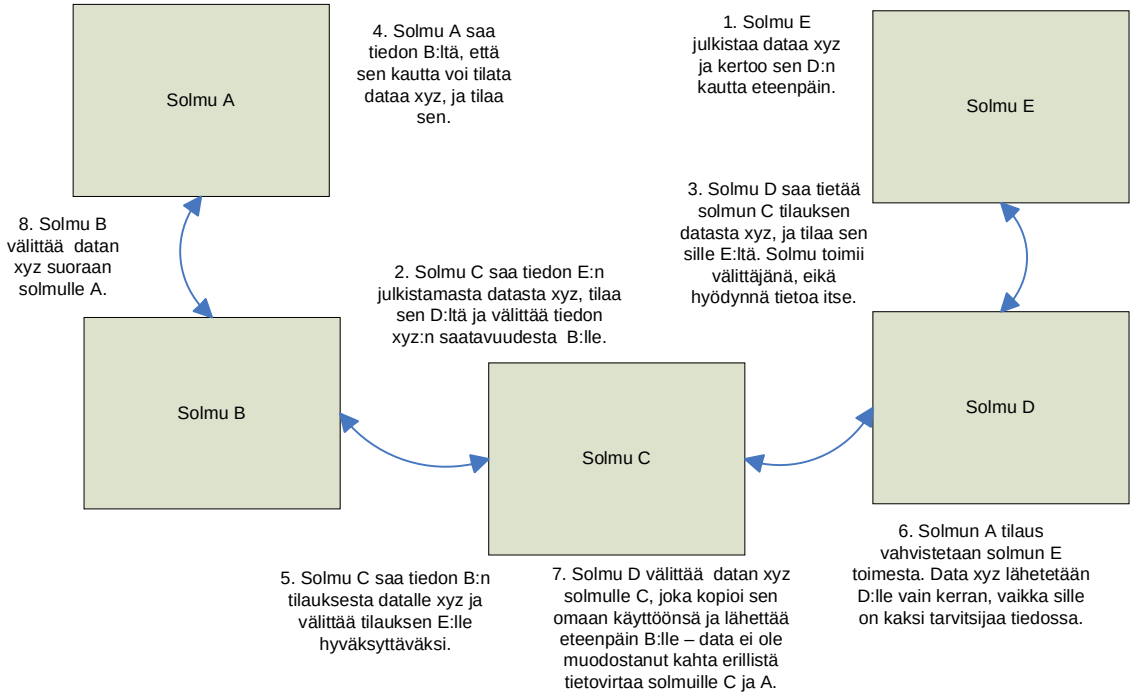
Käyttäen kohdan 2.1 esimerkkiä (kuva 2.1) apuna, tulisi julkaisija–tilaaja-mallin hyödyntäminen muuttamaan toimintaa varsin periaatteellisella tavalla: yhtenäisistä, päästä päähän olevista tietovirroista (end to end connection) tulee välietappisia. Nämä etapit muodostuvat datapalvelimen eri yksiköistä (kuva 3.1). Kuva 2.5 esittää tämän muutoksen graafisesti.

Siinä solmu E julkistaisi solmulle D, että jotain tietoa on tullut saataville sen kautta. Saatavuuden ilmoitus välitettäisiin eteenpäin muille verkon solmuille. Näin solmu C saa tiedon tästä saatavuudesta, ja koska siihen on yhteydessä julkistetun tiedon tarvitsija, ilmoittaa se tarpeestaan D:lle (joka tilaa sen E:ltä ja välittää C:lle). Solmu C ei kuitenkaan pysäytä julkistamisesta kertovaa viestiä, sillä tiedolla voi olla muitakin tarvitsijoita. Solmu C kertoo siis solmulle B, että sen kautta voi saada juuri julkaistua tietoa. Kun tieto saatavuudesta tulee solmulle A asti, tilaa se tiedon itselleen. Tämä tilaus välittyy solmulle C solmun B kautta.

Solmu C ei pysäytä tilauksen kulkemista julkaisijalle asti, vaikka toisenlaiseen käyttötarkoitukseen kehitetyn verkon näin voikin olla hyvä toimia – tiedot tulevat solmulle C jo valmiiksi, joten se voisi vain kopioida ne suoraan uudelle tilaajalle. Syynä tähän on mahdollisuus yksisuuntaisiin reitteihin, jolloin solmulla C ei ole tietoa esimerkiksi E:n kyvystä lähettää tietoja suoraan A:lle. Tämän vuoksi tilaustieto on hyvä välittää tiedon julkaisijalle asti, jos reitityskerros kertoo sen edelleen olevan verkossa – jos se ei ole verkossa, tulee hyväksyntäpäätös kyetä tekemään tilauksen vastaanottaneessa solmussa. Siitä saadaan myös tietoturvallisuuden kannalta hyödyllinen lisäetu: tilaukset voidaan pääosin hyväksyä keskitetysti julkaisijan toimesta.

Tällaisena hyväksyntäviestinä voidaan käyttää joko erikseen hyväksynnästä kertovaa viestiä, tai esimerkiksi tilaajaksi hyväksytyn kohteen julkisella avaimella salattuna välitettävää tietoa tilatun tiedon purkavasta avaimesta.

Koska julkaisija tietää kaikki tiedon tilaajat (ja mitä kautta mikin tilaus on tullut), voi E myös paremmin päätellä, minkä solmun kautta A:lle on parasta välittää tietoja. Tällainen tilausten ketjuttaminen vähentää verkossa välitettävän tiedon määrää huomattavasti, koska tiedolle ei synny useita erillisiä virtoja. Se asettaa myös vaatimuksen hyvästä tietoturvasta, sillä tieto voidaan jättää talteen reitin jokaiselle solmulle. Myös uusintalähetykset voidaan tehdä reitin jostain kohdasta varsinaisen lähettäjän sijaan. Tällöin tiedon ei tarvitse kulkea koko reitin lävitse, vaan vain osan siitä.



Kuva 2.5: Datapalvelimen tuleva toimintamalli

Voimakkaasti salattavan tiedon yhteydessä tilanne monimutkaistuu. Mikäli esimerkin (kuva 2.5) mukaisessa järjestelmässä puhuttaisiin voimakasta salausta vaativasta tiedosta, pitäisi julkaisevan solmun datapalvelimen salata viesti kokonaisuudessaan, ilman sen avaamista lähetystä välitettäessä. Jotta tällöin voidaan hyödyntää verkkoa tehokkaasti, tulee tiedot salata sellaisella avaimella, jonka vain siihen oikeutetut tietävät. Näin tietoa ei tarvitse salata jokaiselle erikseen, vaan sitä voidaan pääasiassa käsitellä kuin mitä tahansa muuta tietoa. Jotta datapalvelimesta saadaan täysi hyöty, tarvitsee sen kuitenkin voida purkaa salaus voidakseen käsitellä tiedon rakennetta ja osia – ilman pääsyä rakenteeseen ja tiedon osiin ei datapalvelin voi yhdistää osatiedon tilausta sille jo valmiiksi tilattuna olevaan tietoon, vaan joutuu muodostamaan siitä erillisen tilauksen.

Tätä tarkoitusta varten voidaan tiedon rakenteita pitää tiedosta erillisinä. Näin tiedon rakenne olisi lähes aina datapalvelimen käytettävissä, ja tiedon osat voidaan salata

erillisinä paloina. Tällöin datapalvelin kykenee yhdistämään osatilauksien tiedot yhdeksi tilaukseksi, vaikka se ei tietäisikään mitään itse tiedoista. Salauksen vuoksi tässä tilanteessa ei kuitenkaan voida koostaa useita tiedon lähetyksiä yhdeksi isommaksi lähetykseksi. Tiedon rakenteisiin ja siirtämiseen liittyvät asiat on selitetty luvussa 5.

Samalla, kun tietopalvelua päivitetään datapalvelimen kautta toimivaksi, päivitetään myös sen tietoturvaa. Päivityksen osasyynä ovat myös järjestelmän hajautuksen mukanaan tuomat uudet haasteet [9, kohta 6.5]. Tietoturvan parantaminen uudessa ympäristössä onnistuu esimerkiksi käyttämällä datapalvelinten yksiköiden välisessä liikenteessä salausta [25, erityisesti kohta 8.6], sekä jakamalla ne eri tietoturvaluokkiin. Näiden luokitusten avulla kunkin datapalvelimen on helppo tarkastaa täyttääkö siihen liittymistä yrittävä solmu ehdot liittymisen sallimiseksi. Samalla voidaan muodostaa erillisiä verkkoja eri luokitusten mukaan, mitä vaaditaan luokiteltuja tietoja käsittelevältä tietojärjestelmältä [25, luku 8].

Käytännössä tämä tarkoittaa esimerkiksi yhteyksistä vastaavalle reitityskerrokselle sitä, että solmujen näkyvyyttä toisilleen tulee voida rajoittaa niille myönnettyjen oikeuksien mukaan. Erillisten verkkojen muodostamisen avulla helpottuu myös datapalvelimen asiakkaiden hallinta, koska oikeudet erittelevät asiakkaita eri datapalvelimen yksiköille. Tätä kautta oikeudet myös rajoittavat muodostuvien verkkojen kokoa, vähentäen tarvetta välittää verkkoa kontrolloivia viestejä isoille joukoille. Erityisesti koko rajoittuu, mikäli eri luokkien välillä toimivat datapalvelimen yksiköt koostavat kyseisten verkkojen tiedot (eli toimivat niin sanotussa domain-tilassa, josta kerrotaan kohdassa 8.4). Jos datapalvelimen tai sen yksikön oma tietoturvaluokitus ei ole riittävä eikä se kykene välittämään viestejä edes salattuna, olisi sen kerrottava syntyneestä virhetilanteesta asiakkuutta haluavalle. Vaikka tieto on tällöin jo päässyt etenemään tarkoitetun joukon ulkopuolelle, voidaan tämän virheilmoituksen avulla tilanteeseen reagoida.

3. TAUSTATIETOA VAATIMUKSIA ASETTAVISTA TEKIJÖISTÄ

Tässä luvussa kerrotaan aluksi datapalvelimen tehtävä sekä tämän jälkeen muita vaatimuksia järjestelmään tuoneista taustatekijöistä. Eri alikohdissa kuvataan myös verkon rakennetta sekä valmiina käyttöön otettavia komponentteja.

3.1. Datapalvelimen tehtävän kuvaus

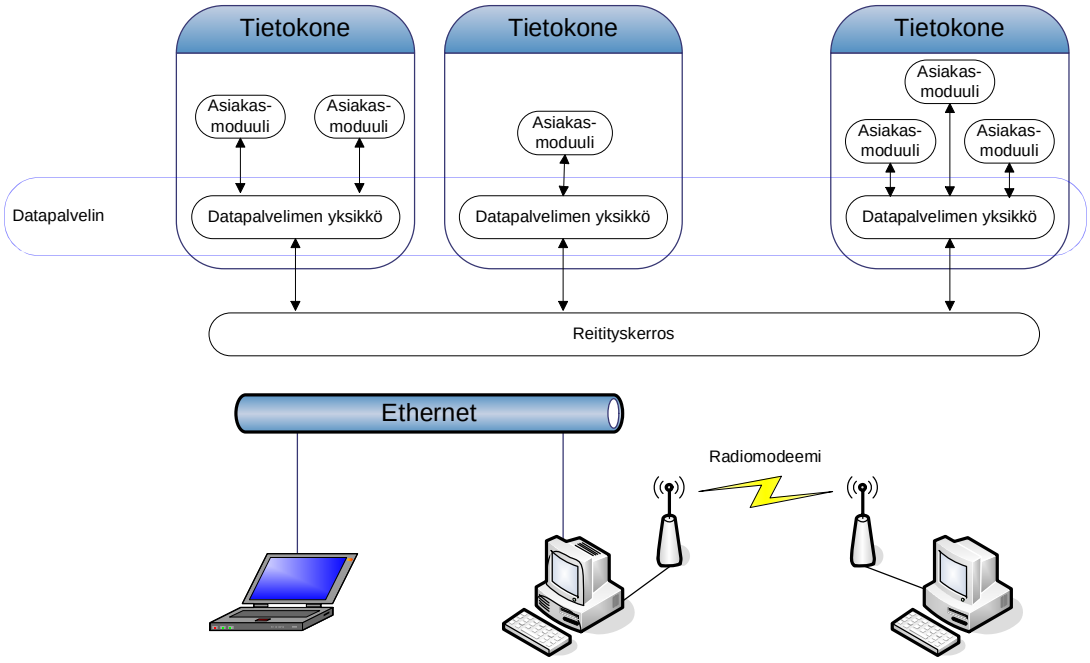
Datapalvelimen varsinainen tehtävä on tietojen jakelu turvallisesti sellaisessa ympäristössä, joka on hajautettu useisiin toimintayksiköihin, ja jonka yksiköiden välisiin tiedonsiirtoyhteyksiin ei voi luottaa. Kuva 3.1 esittää hyvin yksinkertaistetun mallin eräästä mahdollisesta toimintaympäristöstä. Datapalvelimen päätehtävä on jaettavissa kolmeen keskeiseen osatehtävään. Tärkein näistä osatehtävistä on tietojen käsittelyyn liittyvien oikeuksien hallinta ja valvominen. Siksi ohjelmiston vaatimuksiin on sisällytetty Valtiovarainministeriön luottamuksellisen tiedon käsittelyyn liittyvät ohjeet ja vaatimukset [24, 25]. Näihin vaatimuksiin sisältyy muun muassa, että toiminnan pitää noudattaa tunnistamista ohjaavaa AAA-mallia (kohta 2.2.1).

Toinen osatehtävä datapalvelimella on parantaa tietoturvaa, luotettavuutta ja ylläpidettävyyttä poistamalla tiedon tuottajan ja sen kuluttajan tarve tietää toisistaan. Tätä osatehtävää voidaan kutsua myös tilalliseksi erottamiseksi (space decoupling) [5], ja se liittyy kohdassa 2.2.2 esiteltyyn julkaisija–tilaaja-malliin. Sen toteutumiseksi kaikki tietojen vaihto tapahtuu datapalvelimen kautta – näin asiakasmoduulit eivät tiedä edes sitä, onko julkaisun tai tilauksen toinen osapuoli samalla koneella vai verkon kautta yhteydessä johonkin toiseen datapalvelimen yksikköön.

Kolmas osatehtävä datapalvelimella on verkon liikenteen vähentäminen, mikä voidaan saavuttaa yhdistämällä tilauksia keskenään eli ketjuttamalla tilauksia datapalvelimen eri yksiköiden välillä. Näin sama tieto ei kierrä verkossa useaan kertaan. Tehtävä on tärkeä, koska käytettävät tiedonsiirtoyhteydet voivat olla erittäin hitaita ja epäluotettavia, jolloin ne siis voisivat tukkeutua suuresta viestimäärästä erittäin helposti. Verkossa tapahtuvan viestinnän määrä halutaan minimoida myös sovelluskohtaisista tietoturvatarpeista johtuen.

Datapalvelimen yksiköt toimivat verkon solmupisteinä. Ne jakavat tilattuja tietoja sekä sitä kyseiseltä solmulta tilanneille datapalvelimen yksiköille että omille asiakasmoduuleilleen. Datapalvelimen eri yksiköt muodostavat yhtenäisen kokonaisuuden, josta käytetään nimitystä datapalvelin. Kuva 3.1 selventää eroa graafisesti. Datapalvelimen yksiköiden muodostaman kokonaisuuden asiakasmoduulit voivat tuottaa ja kuluttaa määrittelemänsä kaltaista tietoa datapalvelimen kautta. Tietoa

tarpeista ja lähteistä pidetään yllä datapalvelimen sisäisessä tietokannassa eli julkaisu & tilaus -rekisterissä (myöhemmin J&T-rekisteri).



Kuva 3.1: Datapalvelimen ja sen yksikön eron kuvaus

Tämän rekisterin tulee olla yksikkökohtainen, koska eri yksiköiden tiedot voivat erota toisistaan. Tilauksia voidaan ketjuttaa tämän rekisterin avulla, ja sen avulla voidaan myös estää vanhentuneen tiedon lähettämistä. Tällaisen ketjun yhteydessä välitettävät tiedot voivat kulkea mahdollisesti useiden datapalvelimen yksiköiden kautta julkaisijalta tilaajalle, ilman sen näkymistä muille osapuolille. Ketjutuksen avulla muut tietoa tarvitsevat tahot voivat poimia sen ketjun päätepisteiden välissä olevilta solmuilta, eli niille ei tarvitse lähettää omaa tietovirtaa alkuperäiseltä lähteeltä asti. Toisaalta tiedon luottamuksellisuusaste voi rajoittaa mahdollisuuksia tähän – tai jopa estää sen kokonaan, jos datapalvelimen yksikkö ei voi käsitellä tietoa salaamattomassa muodossa.

Verkon jokaisessa solmussa voi olla useita tilaajia ja tuottajia, mutta vain yksi datapalvelin. Reitittävissä solmuissa ei kuitenkaan ole datapalvelinta. Näissä solmuissa sijaitsevat asiakasmoduulit ohjataan liittymään niitä lähinnä olevaan datapalvelimen yksikköön. Tämän ohjauksen suorittaa reitityskerros, eikä sen tulisi näkyä asiakasmoduuleille – edes viiveen merkittävänä kasvuna. Koska kaikki viestinvälitys tapahtuu datapalvelimen kautta, muodostuisi siitä ilman hajautuksen hyödyntämistä yksittäinen kriittinen piste verkkoon.

Kriittisten pisteiden mukana oleminen ei sovi ohjelmistolle asetettuihin tavoitteisiin. Lisäksi ohjelmistoperheen osien pitää voida toimia myös ilman verkkoyhteyttä tai olosuhteissa, joissa verkko voi jakaantua erillisiksi osiksi sekä vastaavasti yhdistyä useasta osasta yhdeksi yhtenäiseksi kokonaisuudeksi. Yksittäisen solmun tulisi myös kyetä toimimaan omillaan, tilanteen näkyessä ainoastaan saatavilla olevan tiedon määrässä. Näiden vaatimuksien toteutumiseksi datapalvelin hajautetaan eri solmuissa

oleviin datapalvelimen yksiköihin. Kuvaus toiminnasta eri yksiköiden välillä löytyy luvusta 8.

Hajautetun datapalvelimen eri yksiköt voivat toimia datapalvelimen muodostavana kokonaisuutena pitämällä yhteyttä toisiinsa verkkoyhteyksillä. Kun datapalvelimen yksikköä käynnistetään, on se aluksi aina muista eristynyt. Tällöin se pyytää reitityskerrosta yhdistämään itsensä lähistöllä oleviin muihin yksiköihin, mutta ei jää odottamaan yhteyksiä vaan aloittaa toimintansa heti. Näin se on toimintakykyinen myös täysin eristyneenä ollessaan, ja vastaavasti yksikön saadessa yhteyden johonkin toiseen datapalvelimen yksikköön, voivat ne liittyä isommaksi kokonaisuudeksi.

Hajautus olisi hyvä tehdä niin, että datapalvelimen kautta tietoa vaihtavien tahojen tarve tietää toisistaan on minimoitu myös datapalvelimen eri yksiköiden välisellä tasolla. Myös hajautettuihin järjestelmiin liittyvät erityishaasteet on otettava erikseen huomioon, eli datapalvelimen yksiköt eivät esimerkiksi saa varata viestintäväyliä itselleen niin, että muut datapalvelimen yksiköt eivät pääse viestimään. Näitä haasteita ja ratkaisuja niihin on kuvattu esimerkiksi Haikalan ja Järvisen teoksessa [9, luku 6].

3.2. Verkko

Tässä kohdassa kerrotaan aluksi vertaisverkkorakenteista yleisesti ja verrataan niitä palvelin–asiakas-malliin. Lopuksi kerrotaan sellaisen verkon rakenne, joka on datapalvelimen suunnittelun pohjana.

3.2.1. Vertaisverkot ja MANET-ympäristöt

Vertaisverkon jokainen solmupiste toimii lähettäjänä, vastaanottajana ja välittäjänä. Tyypillisesti solmu toimii aluksi asiakkaana, ja vastaanotettuaan joitakin tietoja siirtyy osittain myös lähettäjän rooliin tarjoamaan jo vastaanottamiaan tietoja muille. Mikäli jokin muu solmu ei saa suoraan yhteyttä toiseen, voi se mahdollisesti käyttää kolmatta solmua välittäjänä.

Vertaisverkossa ei yleensä joko ole yksittäistä tahoa, jonka kautta kaikki liikenne kulkee, tai sille johtavat tiedonsiirtoyhteydet eivät ole luotettavia. Tämä monimutkaistaa tietoturvallisuuden tuomista vertaisverkkoympäristöön, sillä puhtaissa vertaisverkoissa ei täten ole yksittäistä luotettavaa tahoa, jota voitaisiin hyödyntää autentikoinnin varmistamisessa. Vertaisverkoille tyypilliseksi piirteeksi voidaan luokitella myös jokaisen solmun useat yhteydet verkon muihin solmuihin – ne muodostavat keskenään erittäin tiheästi kytkeytyneen verkon.

Vertaisverkon asiakasohjelmistot toteutetaan yleensä hajautettuna, jotta jokainen verkon solmu voi toimia kaikissa rooleissa. Tällä hetkellä vertaisverkkoja käytetään muun muassa tiedostojen ja tietojen jakamiseen. Tällaisia ovat esimerkiksi Direct Connect-verkko [7, s. 1], BitTorrent-protokolla [16] sekä Skype [1]. Itse verkkoa pyörittävän ohjelmiston päällä toimivilta ohjelmilta ei kuitenkaan vaadita suoranaista hajauttamista. Edellä mainituista Skype tarjoaa verkon yli tapahtuvia puheluita

palveluna. Käyttäen sitä esimerkkinä, sen ääntä nauhoittavaa ja toistavaa osaa ei tarvitse hajauttaa, vaikka muu osa ohjelmistosta on hajautettu.

Mikäli vertaisverkon solmut muodostavat verkon itsenäisesti (ad hoc) ja ne voivat liikkua tai niiden sijainti ei ole ennalta tiedossa, tunnetaan se nimellä MANET (Mobile Ad Hoc NETwork) [15]. MANET:n solmut voivat olla resursseiltaan rajoitettuja, ja yhteystapana käytetään erilaisia langattomia verkkoja. Resursseiltaan rajoittuneita versioita kutsutaan toisinaan erillisellä sensoriverkkojen nimellä. Verkon solmut voivat olla esimerkiksi erilaisia antureita, jotka on levitetty maastoon pudottamalla ne lentokoneesta. Tällöin antureiden on kyettävä muodostamaan verkko itsenäisesti voidakseen jakaa tietoja keskenään ja toimittaakseen ne eteenpäin. Yleistä teoriaa sensoriverkoista sekä esittely eräästä sensoriverkosta löytyy teoksesta [11].

Vertaisverkot ovat erittäin tehokkaita ratkaisuja tällaisiin ympäristöihin. Tärkeä vertaisverkkojen hyödyntämiseen ohjaava tekijä on muun muassa ennalta määrittelemätön solmujen kytkeytymisjärjestys sekä mahdollisesti tuntematon verkon topologia, jolloin palvelin ja asiakas -tyypin hierarkkista verkkoa ei voida luoda etukäteen. Lisäksi yhteyksien muodostamisen epävarmuustekijät ohjaavat hajautettujen ohjelmistojen hyödyntämiseen, koska se vähentää kriittisten solmujen määrää – näin verkon jäsenet kykenevät yleensä suorittamaan omat tehtävänsä, vaikka johonkin tiettyyn solmuun ei kyettäisikään saamaan yhteyttä.

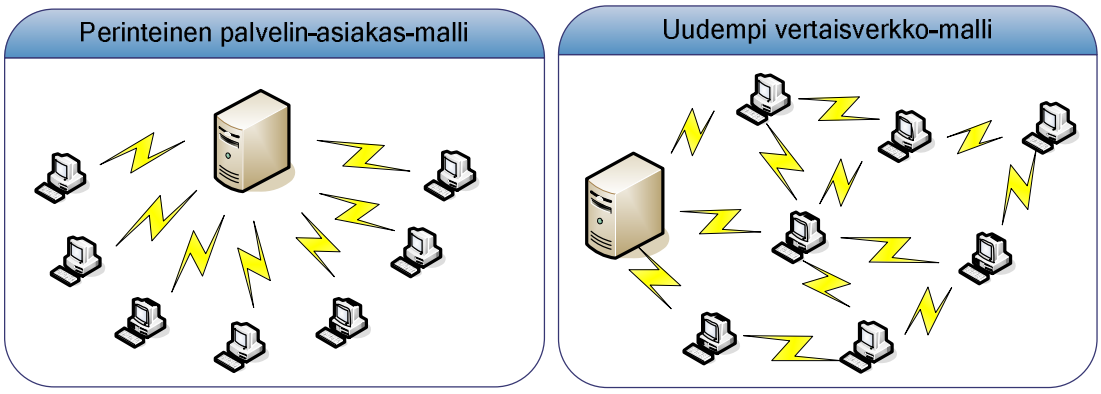
3.2.2. Ero palvelin–asiakas-malliin

Perinteisessä tietojen siirrossa esiintyy palvelin, jolla on tyypillisesti useita asiakkaita. On kuitenkin tehokkaampaa koordinoida tietojen siirtoa niin, että myös asiakkaat kertovat tietoja toisilleen: näin palvelin itsessään ei enää muodostu samanlaiseksi pullonkaulaksi. Tietoa tarvitsevien asiakkaiden välittäessä tietoa toisilleen pääsee tiedon alkuperäinen levittäjä siis vähentämään omaa kaistantarvettaan, koska jokaiselle asiakkaalle ei tarvitse erikseen lähettää samaa tietoa. Lisäksi tästä seuraa että suurempi määrä asiakkaita voi tarkoittaa jopa parempaa toimintaa, kun se perinteisessä mallissa tarkoittaa aina heikentynyttä toimintakykyä – palvelimen käytössä on rajallinen määrä resursseja ja ne pitää jakaa kaikkien asiakkaiden kesken.

Perinteisen ja vertaisverkon tietojen siirtoa on verrattu kuvassa 3.2, missä nuolet kuvaavat tiedon siirtämiseen käytettäviä yhteyksiä. Palvelimeen kohdistuvien yhteyksien määrästä näkee palvelin–asiakas-mallin aiheuttavan suuremman rasituksen palvelimelle jo vähäisellä asiakasmäärällä. Kuvasta ilmenee myös vertaisverkon asiakkaiden kyky hakea tieto kokonaisuudessaan toisilta asiakkailta, ilman suoraa yhteyttä palvelimen kaltaiseen yksikköön. Käytännössä verkossa ei siis ole enää palvelimia ja asiakkaita, vaan pelkästään vertaisasemassa olevia solmuja, jotka ovat sekä asiakkaita että palvelimia toisilleen.

Näin varsinaisen (alkuperäisen) palvelimen ei enää loppuvaiheessa tarvitse olla verkon osana, kunhan tiedot on kokonaisuutena kerrottu verkolle ja sen asiakkaille. Kun joku asiakas saa kaikki tiedon osat haltuun, tulee siitä ikään kuin uusi palvelin entisen

rinnalle. Esimerkiksi kertomalla kullekin palvelimeen yhteydessä olevalle asiakkaalle eri kolmannes jaettavasta tiedosta, palvelimen poistuessa asiakkaat voisivat hakea loput osat toisiltaan.



Kuva 3.2: Palvelin- ja vertaisverkkomallien vertailu

Se ei kuitenkin poista vaaraa tiedon menettämisestä täysin: verkkoon ei ehkä jää pysyvästi ketään kaikkia tietoja omaavaa tahoa – jos jokin noista kolmanneksen vastaanottaneista asiakkaista poistuisi ennen kuin muut asiakkaat saavat sen osuutta tiedoista, tieto kokonaisuutena ei enää olisi saatavilla. Verkon jäsenet voivat kuitenkin saada käsiinsä osittain täydellisen tiedon, mikä voi joissain erityistilanteissa riittää. Mikäli tämä ei ole mahdollista joko tiedon tyypin, laadun tai määrän vuoksi, voi asiakas vaihtoehtoisesti jäädä odottamaan sellaisen tahon palaamista verkkoon, jolla olisi tiedosta puuttuva osa hallussaan. Tällä välin se voi kuitenkin hakea muut osat tiedosta jo valmiiksi.

3.2.3. Oletus datapalvelimen pääasiallisesta verkkorakenteesta

Datapalvelimen yhteydessä käytettävä verkko oletetaan MANET-ympäristön kaltaiseksi niin, että sen solmut eivät ole resursseiltaan rajoitettuja. Sen solmut voivat liikkua paitsi verkossa toimintansa aikana, myös ollessaan kytkeytymättä verkkoon. Verkon rakenteen sallitaan voivan muuttua dynaamisesti, eikä yhteyttä yksittäiseen solmuun taata missään tilanteessa.

Jotta solmut kykenevät toimimaan itsenäisesti tai pienemmissä kokonaisuuksissa verkon osittuessa useaan lohkoon, datapalvelin toteutetaan hajautettuna ohjelmistona. Silloin toisiinsa yhteydet saaneet datapalvelimen yksiköt voivat muodostaa ikään kuin pienemmän datapalvelimen, ja mikäli yhteydet muihin yksiköihin saadaan luotua myöhemmin, voivat nämä pienemmät datapalvelimet yhdistyä isommaksi kokonaisuudeksi. Datapalvelinta käyttäville ohjelmistoille tämä näkyy ainoastaan saatavilla olevan tiedon määrässä.

Verkon toiminnalle lisähaasteita asettaa muun muassa runsas määrä tyypiltään erilaisia viestintään kykeneviä yksiköitä, joiden kaikkien tulee toimia yhdessä. Lisäksi tällainen viestintäyksikkö voi liittyä verkkoon useilla erilaisilla tavoilla – esimerkiksi radiomodeemin, sarjaportin tai lähiverkon kautta. Tämän vuoksi verkossa voi esiintyä

myös yksisuuntaisia yhteyksiä eri tahojen välillä – esimerkiksi radiomodeemeissa käytettävien lähetystehojen erojen vuoksi.

Tässä dokumentissa verkon solmulla tarkoitetaan yleensä verkon viestintään aktiivisesti osallistuvaa yksikköä, jossa yleensä sijaitsee datapalvelimen jokin yksikkö. Esimerkiksi vain reititykseen tarkoitetun solmun tapauksessa datapalvelimen yksikköä ei siinä kuitenkaan ole. Siihen liittyneiden asiakasmoduulien ei kuitenkaan tulisi kyetä havaitsemaan tätä, vaan reitityskerroksen tulisi ohjata yhteydet automaattisesti lähimmälle datapalvelimen yksikölle. Asiakasmoduuleilla ei tulisi olla tarvetta viestiä datapalvelimen lisäksi muille verkon tahoille.

Datapalvelimelle tavoiteltua verkkorakennetta voi myös verrata web-maailmaan. Otettaessa yhteyttä Google:n palveluun, ohjautuvat palvelupyynnöt käyttäjälle sopivalle palvelimelle. Sama osoite ohjaa jopa fyysisesti täysin erillisille palvelimille. Lukuun ottamatta joitakin Internetin sensurointiin osallistuvia maita, toimivat kaikki Googlen palvelimet samojen tietojen pohjalta ja lähes samalla tavalla. Sen palvelimet antavat suuremman prioriteetin hakijan aluetta lähemmille ja samankielisille tietolähteille, mikä tuo oikeastaan ainoan eron niiden välille.

Tämä kuvaa hyvin datapalvelimelta haluttua toiminnallisuutta: eri palvelimilta saadaan pääasiassa samat palvelut. Kuten google.com ohjaa oikealle palvelimelle, tulisi yhteyden luonti oikeaan datapalvelimeen tapahtua verkon rakenteesta ja varsinaisesta yhteystavasta riippumatta. Tämän mahdollistaminen asettaa vaatimuksia datapalvelinta tukevalle reitityskerrokselle, koska ohjaus oikeaan kohteeseen on sen vastuulla.

3.3. Toimintaympäristö muilta osin

Datapalvelin tulee toimimaan ohjelmistoperheessä, josta tämän työn yhteydessä käytetään nimitystä PSVSP. Ohjelmistoperhe toimii voimakkaasti verkottuneessa ja hajautetussa ympäristössä, jossa erityisen kriittistä on sen verkon toimintakunto ja turvallisuus. Tämä ohjelmistoperhe sisältää tuoterungon, jonka ympärille voidaan tehdä uusia toiminnallisuuksia komponenttien avulla. Kyseisen ohjelmistoperheen tuoterunkoa halutaan päivittää tehostamalla sen ytimen muodostavia osia. Eräs tällainen päivitys on tietopalvelun toimenkuvan vahvistaminen muodostamalla siitä datapalvelin.

Itse datapalvelinta suorittava tietokone ei aseteta suorituskykyyn liittyviä rajoitteita. Datapalvelimen voidaan siis katsoa toimivan ilman tiukkoja resurssirajoituksia. Tämä mahdollistaa vahvojen salauksien jatkuvan käytön, sillä erityisesti julkisen avaimen menetelmän hyödyntäminen ei aina onnistu esimerkiksi sensoriverkkojen sensoreiden rajallisten muisti- ja suoritusyksikköresurssien vuoksi [20]. Resursseja hyödynnetään datapalvelimen yksiköissä myös käyttämällä suurta välimuistia, jolloin voidaan säästää verkon kuormituksessa.

PSVSP:n yhteydessä käytettävän verkon ominaisuuksien vuoksi ohjelmistoperheeseen voidaan ottaa alkuvaiheessa käyttöön toiseen ohjelmistoperheeseen kehitetyt OSI-mallin [14, s. 11] mukaiset verkko- (reittien hallinta) ja kuljetuskerrokset (pakettien järjestys ja niiden siirron oikeellisuuden

varmistus). Datapalvelin sijoittuu OSI-mallin rakenteessa näiden yläpuolelle istunto-kerrokseen (kuka lähettää kenelle ja mitä). DOD-malli [14, s. 79] on OSI-malliin verrattuna hyvin samankaltainen, kerrosten nimien vaihtamisen lisäksi siinä on lähinnä yhdistetty kolme ylintä ja kaksi alinta kerrosta omiksi kerroksiksi. Myös DOD-mallissa datapalvelin sijoittuu reitityskerroksen yläpuolelle Process/Application-kerrokseen – tämä kerros vastaa OSI-mallin kolmea ylintä kerrosta. Reitityskerros olisi tässä mallissa jakautunut host-to-host ja internet-kerroksiin, jotka vastaavat OSI-mallin verkko- ja kuljetuskerroksia.

3.4. Yhteistyötahot

Tässä kohdassa kerrotaan lyhyesti neljästä erilaisesta moduulista, jotka otetaan datapalvelimen käyttöön. Niiden toteutus on irrallaan datapalvelimen itsensä toteutuksesta.

3.4.1. Verkkomoduuli eli reitityskerros

PSVSP-ohjelmistoperheessä viestien välityksen toteutukseen on suunniteltu osallistuvan kolme osaa, joista yksi – eli datapalvelin – on tämän työn aihe. Nämä osat jakaantuvat hierarkkisesti kolmelle tasolle, mutta kukin niistä on kuitenkin oma kokonaisuutensa. Näitä osia hyödyntävät asiakasmoduulit tulisivat sijoittumaan datapalvelimen yläpuolelle. Nämä kolme tietojen välityksen ytimen muodostavaa osaa ovat:

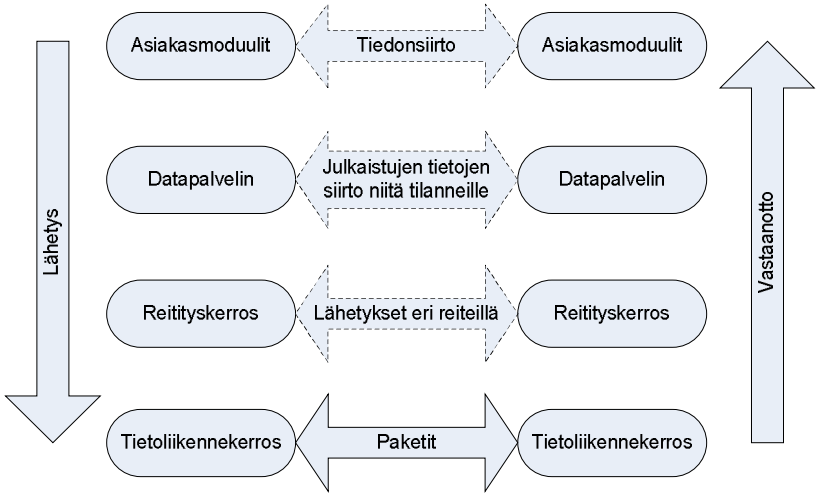
- datapalvelin
- reitityskerros ja
- tietoliikennekerros.

Datapalvelin hyödyntää reitityskerrosta toimissaan, ja vastaavasti reitityskerroksen osat hyödyntävät tietoliikennekerrosta omien toimiensa toteuttamiseksi. Tietoliikennekerros vastaa yksittäisten tietoyhteyksien hallinnasta ja niiden välillä liikkuvasta tiedosta. Tällaista kerrosrakennetta hyödyntämällä alempien kerroksien tarvitsee hoitaa vain omat asiansa, välittämättä siitä mihin niitä käytetään. Ylemmät kerrokset vastaavat omista toimistaan ja tuottavat alempien kerroksien avulla jotain uutta toiminnallisuutta kokonaisuuteen.

Tähän työhön liittyvät kerrokset on esitetty graafisesti (kuva 3.3) niin, että laidoissa olevat nuolet kuvaavat siirtymistä kerrosten välillä ja katkoviivoitetut nuolet tietyn kerroksen tasolla tapahtuvaa liikennettä. Käytännössä ylemmät kerrokset eivät näe alempien kerroksien käyttöä – esimerkiksi datapalvelimen kerroksen tehtävän toteuttaminen voi vaatia useita lähetyksiä ja vastaanottoja alemmilla tasoilla, mutta tämä näkyisi ylemmälle tasolle silti vain yhtenä siirtona datapalvelimen kerrokseen ja kerroksesta.

Tällaisen kerrosjaottelun taustalla on sen suoma luonteva tapa jakaa kokonaistehtävä (viestienvälitys) useisiin eri tehtäviin, jotka kukin voidaan toteuttaa itsenäisinä projekteina. Lisäksi se tarjoaa mahdollisuuden tehostaa muunneltavuutta, jos rajapinnat kerrosten välillä toteutetaan standardien mukaan. Tällöin kunkin kerroksen voidaan

ajatella keskittyvän vain omien tehtäviensä hoitamiseen, ja luottavan muiden kerrosten oikeaan toimintaan. Näin jokainen kerros yksinkertaistuu, mikä helpottaa virheiden löytämistä ja kerroksen toiminnan hallintaa. Samalla kokonaisuudesta ei tule liian monimutkaista esimerkiksi muutosten tekemistä varten. Saadut edut ovat olleet tavoitteena, kun OSI- ja DOD-malleja on kehitetty [14, s. 13].



Kuva 3.3: Kerroksittain tapahtuva tiedon siirtäminen

Tietoliikennekerroksen avaamien yhteyksien päällä toimii reittejä eri pisteiden välillä. Reitityskerroksen tehtäviin kuuluu näiden reittien hallinta ja datapalvelimen tiedottaminen uusista reiteistä. Reitityskerros voi avata uusia reittejä paitsi datapalvelimen pyynnöstä, myös löydettyään itsenäisesti jonkin toimintaa tehostavan reitin. Eräs mahdollisesti käyttöön soveltuva reitityskerros ja tietoliikennekerros on esitelty diplomityössä [26]. Datapalvelimen yhteydessä voidaan käyttää myös muita reitityskerroksia, mikäli ne muodostetaan rajapinnan kanssa yhteensopiviksi.

Datapalvelin voi pyytää reitityskerrosta muodostamaan yhteyden johonkin toiseen solmuun kyetäkseen välittämään viestejä. Tämä pyyntö tulisi voida tehdä esimerkiksi sellaisessa tilanteessa, jossa toimijoiden välillä ei entuudestaan ole saatavilla tarpeet täyttävää yhteyttä esimerkiksi luottamuksellisuusasteen vuoksi. Uuden reitin avautuessa reitityskerros kertoo tästä tapahtumasta yhteyden päissä oleville datapalvelimen yksiköille. Reitin avanneessa päässä oleva yksikkö tekee tarpeidensa mukaiset tilaukset vastapuolelta, joka sen jälkeen kertoo sille omat vastaavat tarpeensa. Jos yksikkö ei kykene täyttämään tarpeita, ilmoittaa se nämä tarpeet muiden yhteyksiensä päissä oleville yksiköille. Näin molemmissa päissä olevat datapalvelimen yksiköt voivat täyttää tilauksiensa tarpeet paremmin, kun tiedot välitetään suuremmasta joukosta asiakasmoduuleita.

3.4.2. Tiedostomoduuli

Tiedostoja hallitsevaa moduulia käytetään kolmeen tarkoitukseen: välimuistin ylläpitoon, asetustiedoston lukemiseen sekä tilannevedoksen tallentamiseen ja lukemiseen. Näihin tehtäviin kykenevä ja tietoturvatarpeet täyttävä moduuli tiedostojen

hallintaan on jo olemassa, ja se voidaan ottaa datapalvelimen käyttöön. Näin datapalvelin voi paremmin keskittyä omien ydintoimintojensa hoitamiseen. Samalla datapalvelin muodostuu selkeämmäksi kokonaisuudeksi, eikä joudu vastaamaan esimerkiksi tiedostojen salauksesta ja turvallisuudesta.

Välimuisti voidaan kirjoittaa datapalvelinta pyörittävän tietokoneen kiintolevylle joko kokonaan tai osittain. Se voidaan myös pitää täysin keskusmuistissa. Kirjoittamalla kuitenkin osa tiedoista pysyväismuistiin voidaan pitää tallessa suurempi määrä tietoja. Välimuistin koko määritellään ohjelmiston käynnistyksen yhteydessä luettavassa asetustiedostossa sekä keskusmuistin että kiintolevyn osalta.

Tilannevedoksesta voidaan myöhemmin palauttaa datapalvelin samaan tilaan kuin se oli sitä tallennettaessa, vaikka sitä ajava tietokone olisikin välillä ajettu alas. Tämän vuoksi tilannevedoksen yhteydessä pitää kaikki muistirakenteet saada kirjoitettua talteen odotettavissa olevan ohjelmiston sammutuksen vuoksi, sillä muuten ohjelmistoa on hankalaa saada samaan tilaan kuin se oli tilannevedosta tehtäessä. Kohta 8.5 käsittelee tilannevedoksen yhteydessä tehtävät toimenpiteet.

3.4.3. Salausmoduuli

Datapalvelimen muodostamissa yhteyksissä tulee olla mahdollista käyttää useita erilaisia salauksia – myös sellaisia, joita ei tätä kirjoitettaessa vielä ole kehitetty. Tämän vuoksi salauksien toteuttaminen on ulkoistettu omaksi moduulikseen – näin sitä on helpompi päivittää. Tämä moduuli on toteutettu muussa yhteydessä ja otetaan täten käyttöön valmiina ratkaisuna.

Sen tarjoamien palveluiden turvallisuus on varmistettu, jolloin moduulia käyttämällä vältetään riski toteuttaa salausprotokollat tietoturvaltaan epätäydellisellä tasolla. Erillisyytensä vuoksi samaa salausmoduulia voidaan hyödyntää myös tiedostoista vastaavassa moduulissa. Itse datapalvelin ja sen yksiköt käyttävät salauksia pääasiassa verkkoyhteyksien päällä välitettävän tiedon turvaamiseen. Vaikka salausmoduuli vastaa itse salausten toteutuksesta, on datapalvelimen vastuulla asettaa oikeat salaukset käyttöön. Tämä tapahtuu välitettävän tiedon perusteella.

Avainten hallinnalla tarkoitetaan salauksissa käytettyjen julkisten, yksityisten ja salaisten avainten hallintaa. Tavoitteena on estää avainten vaarantuminen vihamielisille kohteille, eli se liittyy hyvin vahvasti turvallisuuteen. Kuten tavallisten lukkojen avainten hallinnan kanssa, pitää myös salausten purkamiseen käytettyjen avainten kanssa olla tarkkana. Paraskaan lukko ei auta, jos sen avain on murtovarkaalla, ja sama pätee salauksiin: tieto ei ole suojattuna, jos avain tiedon salauksen purkamiseksi on väärien tahojen saatavilla. Varsinainen avainten hallinta on rajattu työn ulkopuolelle.

Asetustiedostossa määritellään datapalvelimen käyttämät salaukset. Salauksien saatavuus tulee tarkastaa salausmoduulilta ennen niiden käyttöä mahdollisen virhetilanteen estämiseksi. Tällä tavalla datapalvelin voi ottaa salausprotokollia käyttöön solmukohtaisesti, ja mahdollistetaan salauksen käytön estäminen ilman tarvetta poistaa sen toteutus salausmoduulista. Uuden, salausmoduuliin lisätyn

salauksen käyttöön ottaminen tapahtuisi vastaavasti lisäämällä se käytettävien salauksien listaan asetustiedostossa.

Päätös käytettävästä salauksesta riippuu pääasiassa tiedon turvallisuusluokituksesta. Asiakasmoduuleilla on kuitenkin oikeus pyytää jonkin avaimen ja salausmenetelmän käyttämistä tilauksen tietojen välittämiseen. Julkaiseva taho ja matkan varrella kohdattavat datapalvelimen yksiköt voivat kuitenkin hylätä tällaisen pyynnön. Tämä hylkäys voi perustua avaimen epäkelpoisuuteen tai salausmenetelmän riittämättömyyteen. Salausmenetelmä hylätään myös sen toteutuksen puuttuessa julkaisijalta.

3.4.4. Tulkkimoduuli

Tulkkimoduulin tehtävänä on tulkita sen saamat tiedot niille haluttuun muotoon. Tässä tehtävässä hyödynnetään säännöllisiä lausekkeita (regular expression) [13]. Tämän moduulin toteutus on perua aikaisemmasta ohjelmistoperheen versiosta sekä sen tietopalvelusta. Tarkoituksena tulkkimoduulilla on peittää tietorakenteissa tapahtuvat muutokset viestejä vastaanottavilta tahoilta. Tulkkimoduulin hyödyntämisen etuna jotain tiedon kuluttajaa ei tarvitse muuttaa, mikäli saman tiedon tuottaja muuttaa samaisen tiedon rakennetta. Periaatteessa tulkkimoduuli siis pitää yllä datapalvelimen ulkopuolista metatietoa tietojen rakenteen tai esitysmuodon muutoksista.

Tämä tekee siitä erittäin tärkeän moduulin järjestelmän pitkäikäisyyden kannalta. Moduuliin voidaan liittää uusia ohjeita tietojen muuttamiseksi sitä mukaa kun datapalvelimen kautta siirrettävät tiedot muuttuvat. Tämän vuoksi on tarpeen mahdollistaa tietyn tulkkimoduulin version tai osan sitominen johonkin tiettyyn metatietoon liittyviin tilauksiin. Kun tämä mahdollistetaan ja sopivat moduulit sekä tehdään että sidotaan tehtyihin rakenteiden muutoksiin, tulevat tulkkimoduulin kautta pyöräytettävät tiedot aina samassa muodossa vastaanottajalle.

Esimerkiksi tällaisesta tiedon kuvauksen muutoksen tarpeesta voidaan ottaa tilanne, jossa koordinaattitiedoissa on pilkku erottamassa koordinaatit toisistaan. Mikäli tuo pilkku myöhemmin jätetään pois tai korvataan jollain muulla merkillä, pitäisi kaikkia tuota tietoa vastaanottavia komponentteja muuttaa. Hyödyntämällä tulkkimoduulia asiakasmoduuliin yhteydessä olevassa datapalvelimen yksikössä näin ei kuitenkaan tarvitse toimia, sillä se voi tulkata tiedon tarvittaessa sen vanhempaan versioon. Koska muunnos tehdään vasta juuri ennen asiakasmoduulille luovutusta, välittävät kaikki datapalvelimen yksiköt tiedon toisilleen samassa muodossa.

Vastaavasti tietoa on aiemmin voitu tuottaa yhdestä paikasta, joka myöhemmin on voitu jakaa useammaksi tuottajaksi. Tekemällä sopiva lisäys tulkkimoduuliin voi se nyt kerätä ja yhdistää usealle tuottajalle jaetun tiedon takaisin yhdeksi tiedoksi. Esimerkki tällaisesta voisi olla lentokoneen maanopeuden tietoihin kohdistuva muutos: aiemmin se on ollut saatavilla suoraan yhdestä laitteesta, ja myöhemmin se on jaettu erikseen tuulen nopeutta ja todellista lentonopeutta ilmoittaviksi osiksi [6, sivu 9, kohta Ground Speed].

4. TIETOTURVALLISUUS JA SEN HUOMIOINTI

Tässä luvussa kerrotaan aluksi tietoturvallisuudesta yleisesti. Kohdassa 4.2 kuvataan menetelmiä tietojen salaamiseen sekä niihin liittyviä turvallisuusasioita. Kohdassa 4.3 käydään lävitse verkkoon kohdistuvat uhkakuvat ja lopuksi kerrotaan tapoja turvallisuuden huomioimiseen koko ohjelmiston toiminnassa.

4.1. Mitä on tietoturvallisuus

Tietoturvallisuus voidaan yleisesti jakaa kolmeen osaan: eheyden, käytettävyyden ja luottamuksellisuuden hallintaan. Näistä eheydellä tarkoitetaan tietojen muuttamisen estämistä ja havaitsemista. Vastaavasti käytettävyydellä tarkoitetaan sitä, ettei viestin välittämistä perille voida estää ilman sen estäneen tahon tunnistamista vihamieliseksi tahoksi – tämän jälkeen se voidaan ottaa huomioon viestinnässä, esimerkiksi ottamalla käyttöön vaihtoehtoinen reitti tai odottamalla viestinnän estävän vaikutuksen poistumista ennen viestin uudelleenlähetystä. Luottamuksellisuus kuvaa samalla tavalla, että viestiin pääsevät käsiksi vain ne tahot, joille se on tarkoitettu. Listaa voidaan täydentää esimerkiksi kiistämättömyydellä (lähettäjä ei voi kiistää lähettäneensä viestiä) ja tunnistuksella (käyttäjän voidaan luotettavasti todeta joko olevan se taho, joka hän väittää olevansa, tai jokin muu taho). [25, kohdat 4.2.2, 4.2.3, 4.2.4 ja 4.3]

Datapalvelin ei saa päästää luottamuksellista tietoa kulkeutumaan väärille vastaanottajille, joten tietoturvallisuus on viestien varsinaisen välittämisen ohella tärkein osa-alue datapalvelimen toiminnassa. Tietojen eheyden, käytettävyyden ja luottamuksellisuuden tärkeyttä ei voi korostaa liikaa, sillä niiden varmistaminen on datapalvelimen päätehtävä. Tietoturvallisuus suunnitellaan vastaamaan valtiovarainministeriön vaatimuksia luottamuksellisille tiedoille. Esimerkiksi kaikesta luottamuksellisten tietojen käsittelystä ja niihin liittyvistä pyynnöistä on pystyttävä tuottamaan raportti [25 kohta 10.4;24 s. 72 kohta 13].

4.2. Salauksista

Tässä kohdassa kuvataan menetelmiä tietojen salaamiseen sekä niihin liittyviä turvallisuusasioita. Aluksi kerrotaan kaksi yleisesti tunnettua menetelmää tietojen salaamiseen sekä mahdollisuus menetelmien yhdistämisestä. Lopuksi kerrotaan salauksien turvalliseen käyttöön liittyvistä tekijöistä ja suositellaan muutamia tällä hetkellä turvalliseksi luokiteltuja algoritmeja.

4.2.1. Symmetrisen avaimen menetelmä

Symmetrisissä salausmenetelmissä sekä tiedon purkamiseen että salaamiseen käytetään samaa avainta. Nimitys tulee tästä avaimen käytön symmetrisyydestä. Tämän tyypin salaukset ovat olleet ensimmäisiä salausmenetelmiä. Niiden käyttäminen on myös laskennallisesti halvempaa kuin uudempien, epäsymmetristen avainten tapauksessa. Niitä kutsutaankin perinteisiksi salauksiksi [10, s. 3]. Eräs kuuluisa esimerkki symmetrisestä salauksesta on niin kutsuttu Julius Caesarin salaus, jossa viestin kirjain vaihdetaan aakkosissa avaimen verran seuraavaan.

Mitään tällaisia yksinkertaisia kirjaimen vaihtoon suoraan avaimen mukaan pohjautuvia menetelmiä ei enää katsota turvallisiksi, koska ne voidaan purkaa kohtuullisen vähällä vaivalla tietokoneiden avulla – esimerkiksi listaamalla kaikki mahdolliset purkutulokset jostain salatun tekstin lauseesta. Tällaisesta listasta erottuu todennäköisesti vain yksi selväkielisenä, eikä listaan tule suuria määriä rivejä. Myös erilaisia toistuvuuksia voidaan etsiä, jolloin siirroksen vaihtelusta kirjaimen sijainnin mukaan ei ole apua.

Nykyisin symmetrisiin salauksiin käytetään huomattavasti mutkikkaampia menetelmiä, jotka tietokoneiden käyttö on mahdollistanut. Tällaisia menettelyjä ovat muun muassa AES [21, kohta 2.5.3] sekä Khazad [21, kohta 2.4.2]. Ne sisältävät useita erilaisia laskutoimituksia, joita ei ilman tietokonetta ole helppoa suorittaa. Useita tällaisia salausmetodeita on esitelty ja analysoitu NESSIE-ryhmän turvallisuusraportissa [21, s. 7–89].

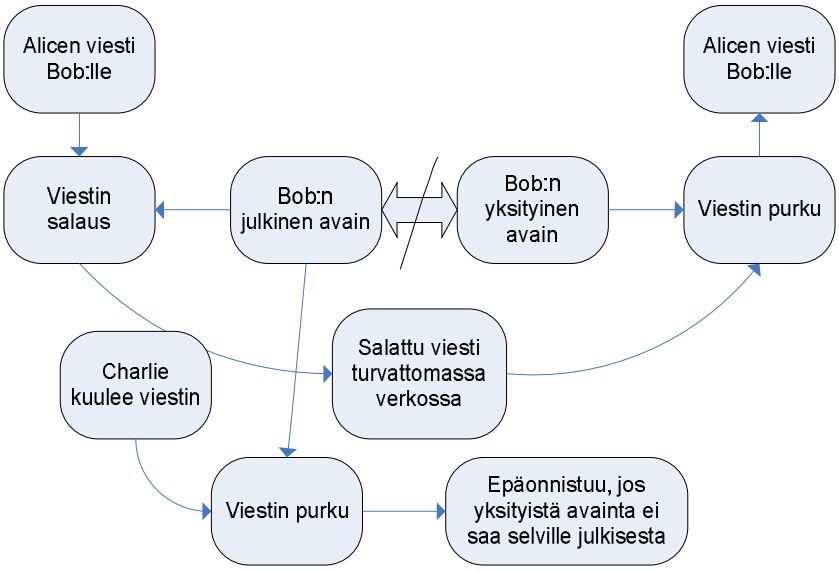
Symmetrisiä avaimia ei voi kertoa verkon yli toiselle osapuolelle, sillä sitä mahdollisesti kuunteleva kolmas osapuoli saisi avaimen samalla käsiinsä. Myös useiden symmetristen avainten muistissa pitäminen aiheuttaa tietoturvariskin – jos kohteen muistissa olevat tiedot saadaan luettua, olisivat kaikki salatut tiedot välittömästi auki. Näiden riskien vuoksi tulisi epäsymmetrisiä salauksia hyödyntää niissä tilanteissa, joissa on pakollista jakaa avaimia verkon ylitse tai pitää niitä muistissa – julkisten avainten paljastuminen ei aiheuta ongelmia.

4.2.2. Epäsymmetrisen avaimen menetelmä

Kuva 4.1 esittää epäsymmetrisen menetelmän käytön pääkohdat. Julkisella avaimella tarkoitetaan sellaista avainta, jolla voidaan salata viesti niin, että sen saa auki vain yksityisellä avaimella. Vastaavasti yksityisellä avaimella allekirjoitetun viestin voi varmistaa tulevan oikealta henkilöltä julkisen avaimen avulla. Symmetrisissä salausmenetelmissä käytetään salaamiseen samaa avainta, ja epäsymmetrisissä salaamiseen käytetään eri avainta kuin salauksen purkamiseen – tämä on niiden välinen, oleellinen ero. [19, luku 8]

Tällainen avainpari tulee luoda niin, ettei yksityistä avainta voi saada laskettua järkevässä ajassa julkisen avaimen pohjalta käytössä olevalla tekniikalla [10, s. 3]. Käytännössä tämän vaatimuksen vuoksi saman tietoturvatason saavuttamiseksi

epäsymmetrisiä menetelmiä käytettäessä tulee avainten olla pidempiä kuin niiden tarvitsee olla symmetristen menetelmien yhteydessä [21, taulukko kohdassa 6.2.3.3]. Autentikoimisen varmistamiseksi pitää lisäksi sitoa julkinen avain tiettyyn varmenteeseen [2, s. 233–234]. Erona symmetrisiin salausmenetelmiin on suurempi rasite, joka johtuu aiemmin mainitusta käytettävien avainten pituuden kasvusta ja menetelmässä käytettävien laskujen monimutkaisuudesta.



Kuva 4.1: Epäsymmetrisen salausmenetelmän käyttäminen

Epäsymmetrisiä salauksia käytettäessä datapalvelimet voisivat kertoa omat julkiset avaimensa toisille datapalvelimille julkaisu & tilaus -rekistereiden vaihtamisen yhteydessä. Tällä tavalla toimittaessa saatetaan kuitenkin avata tietoturvariski: kertomalla oma julkinen avain jonkun toisen julkiseksi avaimeksi voisi saada mahdollisuuden purkaa kohteelle osoitetut viestit. Tämän estäminen on avainten hallinnasta vastaavan tahon vastuulla. Julkisen avaimen luotettavaa kertomista varten on kehitelty useita ratkaisuja, esimerkiksi lähteen [28] mukainen CSRAN-protokolla. Menetelmille tehdyn analysoinnin vähyyden takia niiden käyttöönottoa ei kuitenkaan voi vielä suositella.

4.2.3. Hybridimenetelmät

Hybridimenetelmillä tarkoitetaan toimimista osittain sekä symmetrisillä että epäsymmetrisillä toimintatavoilla [21, kohta 6.3.1]. Toimimalla näin voidaan yhdistää symmetristen menetelmien suhteellinen keveys epäsymmetristen menetelmien epäluotetuissa verkoissa tuomaan turvallisuuteen. Esimerkiksi varsinaisten tilaajien selvittämisen jälkeen julkaiseva datapalvelimen yksikkö voisi salata varsinaiset tiedot uudella, vain tätä tietoa varten luomallaan symmetrisen salauksen avaimella.

Tämä yhteysavain lähetettäisiin epäsymmetrisin menetelmin salattuna hyväksytyille tahoille, kukin viesti salattuna vastaanottajan julkisella avaimella. Kun yhteysavain on toimitettu turvallisesti, voidaan sillä salattujen tietojen lähettäminen aloittaa.

Yhteysavain voidaan myös jakaa etukäteen niin, että yhteysavaimen tietäminen on rajoitettu tiedon käsittelyyn sallituille solmuille – tällöin yhteysavainta ei tarvitse lähettää verkon ylitse lainkaan eli käytetään tavallista symmetristä salausta.

Tämän yhteysavaimen kertomisen jälkeen voidaan sillä salatut tiedot lähettää turvallisesti verkon ylitse: salauksen voivat purkaa vain ne tahot, jotka tietävät juuri sen tiedon yhteysavaimen. Tämä perustuu kahteen mahdolliseen tapaan saada yhteysavain: sen salaus on joko ollut yksilöllistä mahdollisen verkon yli toimituksen yhteydessä, tai se olisi jaettu jollain tietoturvallisella tavalla ilman verkon käyttöä. Se olisi siis vain niillä tahoilla, jotka on hyväksytty käsittelemään noita tietoja. Vaikka tämä avaimen lähetys tapahtuu jokaiselle yksittäiselle tilaajalle erikseen, ei siitä myöskään seuraa liikaa rasitusta verkolle – verrattuna datavirran salaamiseen erikseen jokaiselle vastaanottajalle säästyy siirtokaistaa moninkertaisia määriä.

4.2.4. Varmenteet ja avainten hallinta

Avainten vaihtamisen toteuttaminen eri datapalvelimen yksiköiden välillä tarpeeksi luotettavasti on haastavaa, koska käyttöympäristö on hajautettu. Erilaisia vaihtoehtoja avainpalvelun hajauttamiseksi on kehitelty useita (muun muassa [17]), mutta niiden turvallisuutta ei kuitenkaan ole luotettavasti todistettu PSVSP:n edellyttämälle tasolle. Koska voimme olettaa verkon luomisen tapahtuvan suunnitellusti, voidaan esimerkiksi käyttää avainten hallinnasta vastaavaa, verkosta poissa olevaa tahoa luotettavana kolmantena tahona – tämä ei estä muista eristynyttä solmujoukkoa toimimasta. Ongelma on käytännössä siis siirretty toisaalla ratkaistavaksi.

Avainten hallinnan toteuttaa koko järjestelmän ulkopuolinen taho. Kyseisen tahon vastuulla on jakaa tiedon purkamiseen soveltuvat avaimet kaikille tahoille, sekä ohjeistaa niitä kulloinkin käytettävästä avaimesta. Jälkimmäisellä tarkoitetaan esimerkiksi avainten hylkäämisestä ja avainryhmän seuraavaan siirtymisen käskyn antamista, samoin kuin ohjaamista sen suhteen, mitä tietoa salataan milläkin avaimella. Avaimista vastaava taho voidaan käsittää luotettavaksi tahoksi, joten sen myöntämiin varmenteisiin voidaan luottaa.

Tämä luotettu taho joko luo avainparit itse tai varmistaa niiden oikeellisuuden esimerkiksi välittämällä jokin ilmoitetulla julkisella avaimella salattu numero ja odottamalla vastaukseksi samaa lukua allekirjoitettuna tahon yksityisellä avaimella. Näin molemmat avaimet voidaan varmistaa tahon omiksi – mutta tahon henkilöllisyydestä on varmistuttava jotenkin muuten. Tässä vaiheessa yleensä käytetäänkin luotettuja verkkoja tai suoria yhteyksiä. Kun avainten oikeellisuudesta on varmistuttu tai ne on turvallisesti välitetty kohteelle, varmentaa luotettu taho epäsymmetrisissä menetelmissä käytetyt julkiset avaimet. Tämä varmentaminen sitoo julkisen avaimen tiettyyn henkilöllisyyteen. Tästä todisteena käytetään varmentavan tahon allekirjoitusta, jonka tekeminen vaatii luotetun tahon yksityisen avaimen tietämistä.

Varmenteessa voidaan myös ilmoittaa käyttäjään liittyviä muita tietoja, kuten esimerkiksi minkälaisen tietoturvatason tietoihin taholla on oikeus. Tällä varmenteella voidaan siis todistaa, että tietty julkinen avain kuuluu jollekin tietylle taholle, ja mitkä kyseisen tahon oikeudet ovat. Näin ei voida epäsymmetrisiä menetelmiä käytettäessä helposti esittää omaa julkista avainta jonkin toisen tahon julkisena avaimena, sillä se vaatisi myös varmenteet luovan tahon yksityisen avaimen tietämistä – muuten allekirjoitus ei enää vastaa viestiä.

Varmenteet voidaan siirtää verkon ylitse, sillä ne on varmennettu luotettavan tahon allekirjoituksella. Alkuvaiheessa kaikki varmenteet ja julkiset avaimet kannattaa säilyttää, koska keskusmuistia ei katsota rajalliseksi resurssiksi. Sitä oletetaan olevan käytettävissä useita satoja megatavuja yksittäisen varmenteen viedessä tähän verrattuna hyvin vähän muistia (kilotavuja). Näin vältetään tarvetta siirtää varmenteita muistin suhteen huomattavasti rajallisemmin resursseja sisältävässä verkossa.

Jatkokehityksessä voidaan harkita tallessa pidettävien varmenteiden määrän pienentämistä, jos käyttöä halutaan laajentaa käytettävissä olevilta resursseiltaan rajoitetumpiin ympäristöihin. Eräs vaihtoehto tähän on esitelty teoksessa [3]. Tallessa pidettävien varmenteiden määrän vähentyessä yhteyden saannin todennäköisyys verkon eri solmujen välillä tulisi kuitenkin laskemaan, koska solmulla ei välttämättä ole yhteyksien päässä olevien solmujensa varmenteita eivätkä solmut siis voi viestiä turvallisesti keskenään. Tiheissä verkoissa tästä ei kuitenkaan tule ongelmaa, vaikka tiedon käyttämä reitti voikin olla pidempi ja täten aiheuttaa ylimääräistä liikennettä verkossa.

4.2.5. Turvallisuussyistä käyttöön suositeltavat algoritmit

Tarkastussummien laskentaan ja tietojen salaukseen tulee käyttää vain turvallisiksi todettuja algoritmejä. Niiden on oltava turvallisia keskipitkällä aikavälillä (5–10 vuotta), jonka jälkeen voidaan odottaa uutta versiota ohjelmistoperheestä – niitä voidaan tarvittaessa muuttaa päivityksen yhteydessä. EU:n NESSIE-tutkimusryhmän mukaan turvallisia algoritmejä ovat esimerkiksi SHA256 sekä Whirlpool tarkastussummien laskentaan ja AES, Camellia, PSEC-KEM sekä ACE-KEM salaukseen [21, kohdat 4.4.3.1, 4.4.1, 2.5.3, 2.5.1, 6.4.5 ja 6.4.1 vastaavasti].

Näistä AES-salauksen vahvempiin muotoihin (192- ja 256-bittiä) on tuoreiden tutkimusten mukaan löydetty teoreettinen hyökkäys, joka ei kuitenkaan vielä ole käytännössä vaarantanut kumpaakaan salausta [2]. Se on kuitenkin laskenut niiden turvamarginaalia reilusti, minkä vuoksi voi olla perusteltua siirtyä käyttämään muita salausalgoritmejä. Myös SHA-2 tarkastussummien laskenta-algoritmin eri bittiversioiden osalta on löydetty mahdollisia heikkouksia, ja Yhdysvallat onkin kehittämässä siitä versiota 3 [4]. Kun SHA-3 on valittu laadituista ehdotuksista, tulisi mahdollisuus sen käyttöön ottamisesta selvittää erikseen.

Mainitut tarkastussumma- ja salausmenetelmät on tarkoitettu tarkastussummien laskentaan ja määrämittaisten viestien salaukseen (sekä salaisella avaimella että julkisen

ja yksityisen -avaimen periaatteella). NESSIE-ryhmän mukaan on yleistä, että julkisen avaimen avulla salattaisiin käytetty symmetrinen salausavain, ja että varsinainen tieto salattaisiin tällaisella salaisella avaimella tehokkuuden parantamiseksi [21, kohta 6.3]. Tehokkuusetu saavutetaan symmetristen salausmenetelmien laskettavuuden helpottumisen kautta. NESSIE-ryhmä on tutkinut myös Leviathan-salausta, eikä löytänyt siitä sen turvallisuutta vaarantavia virheitä [27]. Kyseinen salaus on suunniteltu jatkuvien tietovirtojen salaamiseen, eli esimerkiksi suojaamaan videokuvaa välittävän tietovirran.

4.3. Verkkoon kohdistuvat uhkakuvat

Tässä kohdassa kerrotaan aluksi yleisellä tasolla datapalvelimen uhkakuvista. Sitten käydään läpi kolme merkittävintä uhkakuvaa, kertoen jokaisen kohdalla miten riski voidaan ottaa huomioon datapalvelimen toiminnassa.

4.3.1. Yleisellä tasolla

Verkossa voi olla mukana tahoja, joille verkon liikenteen ei haluta paljastuvan. Tällaisten ei-toivottujen tahojen tunnistaminen on erityisen vaikeaa verkon dynaamisuuden ja solmujen liikkuvuuden vuoksi. Niiden oletetaan voivan myös viestiä järjestelmään. Koska verkossa tapahtuva liikenne salataan, ei verkon passiivisen kuuntelun katsota olevan uhkana.

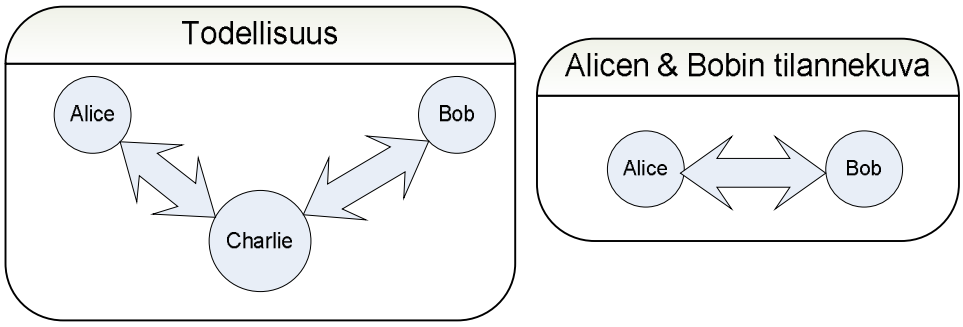
Jos verkkoa vastaan hyökkäävän tahon oletetaan tietävän datapalvelimen toimintamallin eli omaavan mahdollisuuden liittyä verkkoon, voimme havaita kolme varsinaista uhkakuvaa: mies välissä -hyökkäys (tunnetaan yleisesti man-in-the-middle eli MITM-hyökkäyksenä), palvelun saatavuuden estävä hyökkäys sekä datapalvelimen yksikön haltuunotto. Haltuunotto aiheuttaa tietojen luottamuksellisuuden menetyksen ja voi mahdollistaa verkon sisältä tapahtuvan palvelunestohyökkäyksen.

Varsinainen avainten hallinta on kuitenkin toteutettu tämän työn ulkopuolella, samoin kuin muu suojautuminen esimerkiksi haltuunottoa vastaan. Näin ollen tämä uhkakuva ei ole varsinainen uhka datapalvelimen itsensä kannalta, mutta sen olemassaolo on syytä huomioida. Käytännössä datapalvelin luottaa avaintenhallinnan rajoittavan avaimet vain oikeellisille tahoille.

4.3.2. Uhkakuva: mies välissä -hyökkäys

Kuva 4.2 selventää miten MITM-hyökkäyksessä verkkoa vastaan hyökkäävä taho Charlie menee toimijoiden Alice ja Bob verkkoliikenteen väliin, ja teeskentelee Alicelle olevansa Bob ja Bob:lle olevansa Alice. Hyökkääjä haluaa tyypillisesti seurata liikennettä, ja joissain tilanteissa muuttaa viestejä. Julkisuudessa on ollut verkkopankkien tunnuksien hankintaan pyrkiviä, tällä tavalla toteutettuja hyökkäyksiä. Niissä verkkopankin sivun sijaan käyttäjä ohjataan jollekin toiselle sivulle, joka teeskentelee olevansa verkkopankin sivusto. Hyökkäystä ei kuitenkaan voida suorittaa,

mikäli tahot ovat vaihtaneet avaimia tai varmenteita etukäteen – eli hyökkääjän pitää asettaa itsensä tahojen verkkoliikenteen väliin ennen salatun yhteyden avaamista.



Kuva 4.2: MITM-hyökkäyksen vaikutus

Mikäli avaimia ei ole etukäteen vaihdettu eikä yhteys ole jo auki, tapahtuu hyökkäys seuraavasti: Charlie menee Alicen ja Bob:n verkkoliikenteen väliin. Kun Alice haluaa avata esimerkiksi SSH-väylän Bob:lle, estää Charlie viestin kulkeutumisen Bob:lle ja vastaa Alicelle olevansa Bob. Charlie ilmoittaa oman julkisen avaimensa Bob:n julkiseksi avaimeksi, jolloin hänellä on mahdollisuus päästä käsiksi salaamattomaan tietoon. Samanaikaisesti Charlie avaa vastaavan väylän Bob:lle, kertoen tälle olevansa Alice ja vastaavasti välittää oman julkisen avaimensa Alicen avaimena. Jotta Alice ja Bob eivät epäilisi mitään, voi Charlie välittää Alicelta saamansa viestit Bob:lle ja päinvastoin. Koska Charlie saa viestit purettua, voi hän tarvittaessa muokata viestejä tai poistaa niitä valikoiden. Tällaista hyökkäystä on erittäin vaikea tunnistaa sen toteutumisen jälkeen. [21, kohta 8.2.7]

Käytännössä hyökkääjä saa MITM-hyökkäyksen avulla solmujen välisen liikenteen käsiinsä salaamattomana, minkä vuoksi se on erityisen suuri uhkakuva PSVSP:n päälle toteutettavaksi suunnitelluille ohjelmistoille. Eräs tapa vaikeuttaa tällaisen hyökkäyksen toteuttamista on käyttää varmenteita autentikoinnin apuna. Näin voidaan autentikoinnin varmuutta kasvattaa. Tämä on tärkeää, sillä ilman toimivaa autentikointia toisen osapuolen henkilöllisyydestä ei voi varmistua. Autentikointi muodostuu erityisen vaikeaksi vertaisverkoissa niiden usein dynaamisen luonteen vuoksi – niissä ei usein ole varmaa yhteyttä palvelimeen, jolta toisen osapuolen henkilöllisyyden tai varmenteen voimassaolon voisi varmistaa.

Hyökkäyksen toteuttaminen voidaan tehdä vaikeammaksi käyttämällä luotetun tahon myöntämiä varmenteita ja sitomalla niihin varmennetun tahon avain sekä oikeuksiin liittyviä tietoja. PSVSP:n käyttöympäristössä solmujen on todennäköisesti kerrottava toisilleen omista oikeuksistaan. Toisen solmun ilmoitukseen ei voida suoraan luottaa – muutenhan hyökkäävä taho vain kertoisi omaavansa kaikki oikeudet. Tämän vuoksi tarvitaan varmenteen tuomaa luottamusta jonkin tahon kertomiin oikeuksiin.

4.3.3. Uhkakuva: palvelunestohyökkäys

Palvelunestohyökkäyksellä käsitetään yleensä tarkoitettavan hyökkäystä verkon tarjoamien palveluiden estämiseksi. Tämä voi tapahtua pääasiassa kahdella erilaisella

tavalla: estämällä yhteys solmujen välillä kokonaan (esimerkiksi katkaisemalla tietojen siirtämiseen käytetty kaapeli) tai estämällä yhteydenotto palveluun ylikuormituksella.

Vaikka verkon toimijat voivat ylikuormittaa palvelun helposti lähettämällä runsain määrin tietointensiivisiä, vääriä julkaisuja ja tilauksia, ei sen katsota olevan suuri vaara datapalvelimelle. Mikäli jokin datapalvelimen yksikkö saa useita epäonnistuneita tilauksia joltakin taholta, voi se pyytää reitityskerrosta eristämään niitä pyytävän tahon. Verkko rajaisi tällaisen vihamielisen tahon ulkopuolelleen nopeasti, koska myös muut datapalvelimen yksiköt saisivat tällaiselta haitalliselta taholta paljon epäonnistuvia tilauksia tai julkaisuja – joko välitettyinä viesteinä tai vihamielisen tahon siirtyessä välittämään viestejä niiden kautta.

Verkko ei kuitenkaan voi luottaa tällaiseen eristyspyyntöönkään suoraan – muuten se tarjoaa mahdollisuuden hyökätä itseään vastaan pyytämällä eristämään kriittiset tahot. Eräs keino saada varmuus pyynnön alkuperästä on vaatia useita pyyntöjä ennen kuin taho suljetaan pois verkosta. On myös varmistuttava, ettei yksittäinen taho ole useiden pyyntöjen takana. Eräänä keinona tähän on pyynnön allekirjoitus, jonka lisäksi voidaan antaa eristystä pyytävälle taholle työläs laskutehtävä. Jälkimmäisen avulla voidaan myös estää turhien julkaisujen ja tilausten hyväksymistä, koska tällöin hyökkäyksen suorittamisen kustannus on suurempi kuin verkon siitä kärsimä kuormitus.

Suurempi vaara datapalvelimen hyötykäytölle tuleekin yhteyksien estämisestä kriittistä tietoa julkaisevaan datapalvelimen yksikköön. Tällaisia estomahdollisuuksia olisi esimerkiksi radioiden yhteydessä datapalvelimissa käytetyn siirtokaistan tukkiminen sellaisella häiritsevällä lähetteellä, joka estää verkon oman viestinnän kyseistä siirtotietä pitkin. Verkon rakenne pyrkii kuitenkin ohjaamaan liikenteen toista kautta jonkin reitin estyessä. Vaikka tämä ei aina onnistuisikaan, antaa hyvin suoritettu toimintojen hajauttaminen jokaiseen solmuun (ja tätä kautta kriittisten kohteiden määrän vähentäminen) parhaan turvan tällaista hyökkäystä vastaan. Lisäksi välimuistia käyttämällä voidaan prioriteetiltaan tärkeimmät viestit pitää tallessa yhteyksien toimintakuntoon palaamiseen asti.

4.3.4. Uhkakuva: haltuunotto

Verkossa toimivat datapalvelimet oletetaan suojatuiksi suoraa haltuunottoa vastaan. Vaikka tämä on vakava uhka, datapalvelin ei voi suoraan varautua sitä vastaan. Käyttämällä epäsymmetrisiä salausmenetelmiä ja pitämällä tiedossa mahdollisimman vähän arkaluonteista tietoa ja vain julkisia avaimia voidaan kuitenkin rajoittaa haltuunoton aiheuttamia vahinkoja. Näin korkeintaan solmulle itselleen lähetetyt viestit voidaan saada auki haltuunoton yhteydessä.

Ratkaistavaksi jääkin vielä mahdollisuus tilata ylimääräisiä tietoja haltuunoton jälkeen. Tähän voitaisiin jossain määrin puuttua sillä, että muut datapalvelimen yksiköt voisivat tarvittaessa sopia jonkin datapalvelimen yksikön avaimen hylkäämisestä. Tämän jälkeen kyseinen yksikkö tippuisi automaattisesti verkosta, ja se voisi päästä sinne takaisin ainoastaan sen jälkeen, kun luotettava kolmas taho on luonut sille uuden

avainparin. Toinen vaihtoehto on vaihtaa avainpareja säännöllisesti, jolloin hyökkäykseen käytettävissä oleva aika vähenee – solmulle ei toimitettaisi uutta avainparia, jos se on otettu haltuun. Helpointa tämä olisi toteuttaa suorittamalla avainten vaihto jotakin tietoverkosta erillistä, luotettavaa kommunikaatiomenetelmää hyödyntäen. Vastuu tästä avaintenhallinnan osa-alueesta ei kuitenkaan ole datapalvelimella, vaan sen ulkopuolisella taholla.

4.4. Turvallisuuden huomiointi toiminnassa

Tämä kohta kertoo aluksi turvallisuuden huomioinnista normaalissa toiminnassa. Lopuksi esitellään myös turvallisuuden tärkeyden esille tuoma erikoistapaus, sekä kuinka siinä tilanteessa tulisi toimia.

4.4.1. Normaali toiminta

Toisen osapuolen luotettavaan tunnistamiseen liittyy yleensä luotettavan kolmannen osapuolen tekemä varmennus. PSVSP:n käyttämässä verkkoympäristössä tähän ei kuitenkaan ole taattua yhteyttä. Tilanne on kuitenkin otettu huomioon työn ulkopuolella olevassa avaintenhallinnassa. Tunnistamisen jälkeen voidaan tarkastaa sallitaanko jokin toiminta, ja onko toiminta tarpeeksi turvallista tiedon luottamuksellisuusasteen mukaan.

Tiedon käsittelyn turvallisuutta harkitessa ei riitä pelkkä salausalgoritmin tason tarkastelu, vaan varmuuden saamiseksi myös salaamiseen käytettävä avain pitää hyväksyä. Nämä tarkastelut ovat osin eri tasoissa: avaimen tarkastus liittyy osin luotettavaan tunnistamiseen, ja salauksen hyväksyntä taas liittyy tiedon vaatimuksiin sen turvaamisesta. Toisaalta avaimen tarkastamisella voidaan varmistaa, että tietoa ei salata heikolla avaimella.

Tieto on voitu myös jakaa ryhmiin asiakasmoduulien ryhmätunnusten mukaan. Sama tieto voidaan ohjata kahdelle eri ryhmälle erilaisilla tietoturvaan liittyvillä rajoitteilla. Tällaisen ryhmäjaottelun avulla on helpompaa kontrolloida oikeuksia tietoon kuin pelkillä yksittäiseen tunnukseen liittyvillä oikeuksilla – ryhmiä käytettäessä jokaiselle yksittäiselle käyttäjälle ei tarvitse määritellä erikseen oikeuksia tietoon, vaan oikeudet voidaan yleistää tiettyjen käyttäjäryhmien mukaan. Tällöin ei enää ole kyse vain tiedon purkamisen mahdollisuudesta, vaan myös sen käsittelyn oikeuksista.

Tilauksen yhteydessä asiakasmoduuli voi kertoa paitsi käyttöön haluamansa salauksen, myös siihen käytettävän julkisen avaimen. Tilaajan ei ole siis aina käytettävä samaa avainta jokaiselle tilaamalleen julkaisulle. Julkaisija kuitenkin päättää, hyväksyykö se ne. Tieto käytettävästä salauksesta ja julkisesta avaimesta kulkee tilauksen mukana. Datapalvelimen yksiköt pyrkivät käyttämään tilauksissa omia julkisia avaimiaan ja sopivaa salausta.

Datapalvelimen yksikön saadessa viestin, purkaa se salauksen, jos sillä on viestin sisältöön oikeus. Tämän jälkeen se salaa viestin uudelleen, mutta käyttäen tilanneiden tahojen ilmoittamia salauksia sekä avaimia. Uudelleen salaamisen jälkeen viesti

lähetetään sille merkittyihin reitteihin erillisinä viesteinä. Mikäli julkaisija ei hyväksy datapalvelimen yksikön omia avaimia, voidaan tilaus yrittää uusia asiakasmoduulin omilla avaimilla. Näin toimimalla saadaan suurin osa salattua tietoa sisältävistä tilauksista ketjutettua.

4.4.2. Erikoistapaus: datapalvelimelta salattu tieto

On siis mahdollista, että datapalvelimen yksiköllä itsellään ei ole oikeutta käsitellä jotain tietoa salaamattomana. On myös erikoistilanteita, joissa datapalvelimen yksiköllä ei ole oikeutta edes tiedon rakenteeseen. Se voi silti olla sallittua joko joillekin sen omista asiakasmoduuleista tai siihen liitoksissa olevilta datapalvelimen muilta yksiköiltä. Tällöin datapalvelin toimii ainoastaan yksinkertaisena välittäjänä, joka välittämisen lisäksi vain kirjaa lokiin tiedon viestin kulkemisesta sen kautta.

Koska tiedon rakenteesta ei tiedetä, ei tässä tilanteessa datapalvelimen yksikkö siis voi yhdistää tilauksia, ja tiedot voivat kulkea samaa reittiä kahteen kertaan. Nämä salatut tiedot on tällöin salattu erillisillä avaimilla eikä kyseinen yksikkö tiedä tiedon tunnistetta, vaan käsittelee sitä pelkkänä datapakettina kohteelle. Jos kyseessä on datapalvelimen yksikköön suoraan liittynyt asiakasmoduuli, joka välittää salatun pyynnön jollekin toiselle asiakasmoduulille, ei mikään datapalvelimen yksikkö voi tässä tilanteessa suorittaa tiedoille mitään välittämisestä poikkeavia toimintoja.

Tällaisessa tilanteessa kaikki tiedot ohjataan suoraan eteenpäin kohti kohdetta, eikä tilausta voida ketjuttaa. Datapalvelimen yksiköt eivät tällöin voi tarkastaa tiedon esittämiä turvallisuusvaatimuksia, joten niiden täytyy pystyä luottamaan kohteella olevan oikeus ottaa tieto vastaan.

5. TIETOJEN RAKENNE JA SIIRTOTAPA

PSVSP-ohjelmistoperheessä tietoa on kahta muotoa: varsinaista tietoa sekä tietoa kuvaavaa tietoa eli metatietoa. Molemmat näistä esitellään omissa alikohdissa (5.1 ja 5.2 vastaavasti) ja niiden eroja on selvennetty kuvassa 5.1. Selkeyden vuoksi näihin kohtiin liittyviä puita kuvataan XML-kuvauskielellä (eXtensible Markup Language) [23]. Se tarjoaa helpon tavan ilmaista tietojen rakennetta sekä puumaisia kokonaisuuksia tekstimuodossa. Tämän jälkeen kerrotaan tietojen siirtämiselle asetetut vaatimukset sekä kaksi keinoa vähentää verkossa liikkuvan informaation määrää. Viimeisessä kohdassa kerrotaan siirtämiseen käytettävistä menetelmistä.

5.1. Varsinaiset tiedot

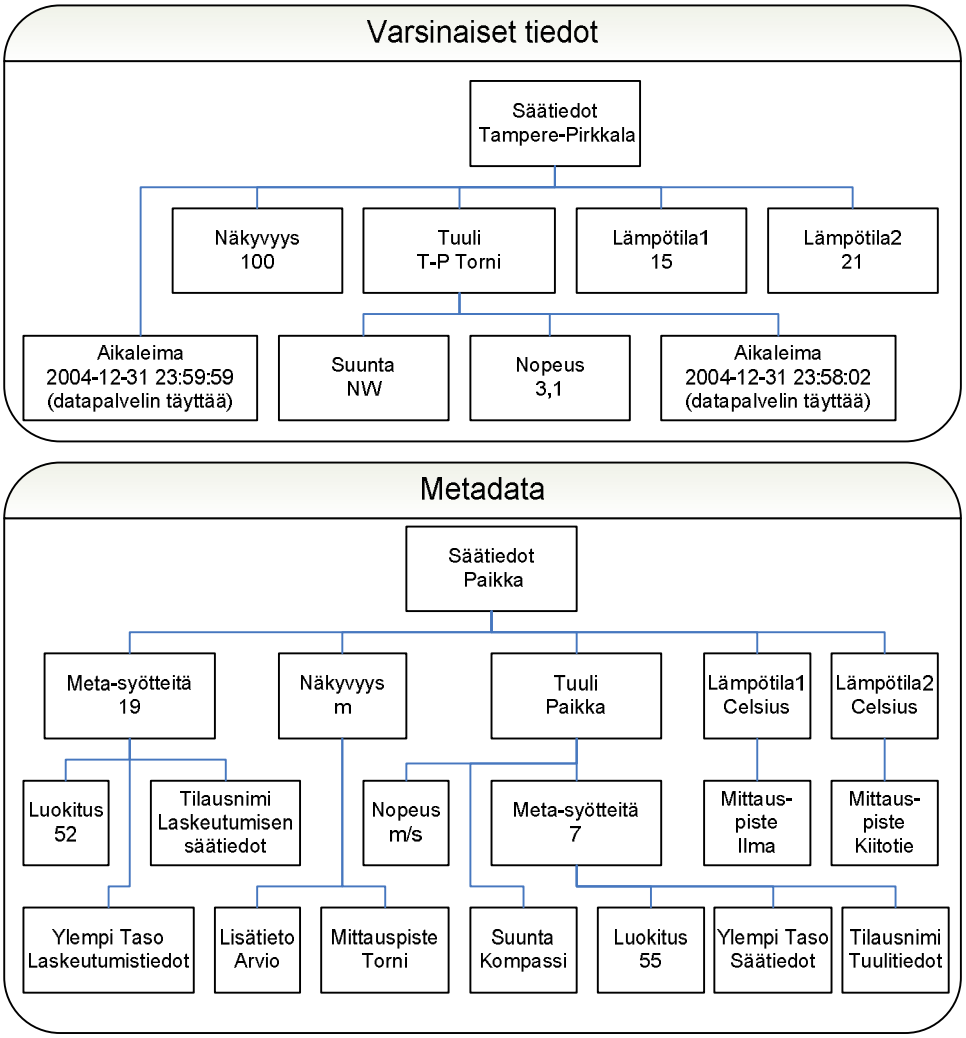
Tieto on tämän dokumentin yhteydessä pari {avain, arvo}. Siihen kuuluu siis varsinaisen tiedon itsensä lisäksi myös tiedolle annettu nimi. Esimerkiksi lämpötilan tieto esitettäisiin siis parina {”Lämpötila”, 12}. Ainoat PSVSP:n asettamat vaatimukset tällaisille avain ja arvo -pareille ovat seuraavat: niiden pitää olla kuvattu metatietojen avulla, ja niiden pitää olla esitettävissä tuoterungon omilla, standardoiduilla tietotyypeillä. Muita vaatimuksia ei syötteelle ole asetettu.

Tiedot ovat yleensä usein lähetettäviä tietoja, kuten esimerkiksi paikkatietoa – jonka lähetyksistä vain tuoreimmasta ollaan yleensä kiinnostuneita. Tiedot voivat myös olla kertaluonteisia, esimerkkinä yksittäinen viesti: todennäköisesti viestin lähettäjä ei halua myöhemmin lähettää samaa viestiä uudestaan. Vaikka samaa viestiä ei toistettaisi, on viesteissä kuitenkin yleensä sama rakenne: lähettäjä, vastaanottaja, otsikko ja itse viesti. Molemmilla esitetyistä tiedon tyypeistä on yhteinen tekijä: samassa muodossa olevaa tietoa odotetaan vastaanotettavan lisää myöhemmin joko samalta tai eri lähteeltä.

Graafinen esimerkki (kuva 5.1) selventää datapalvelimen yhteydessä olevien tietojen puumaisuutta. Käytön edistämiseksi puun ylimmäisen tietokentän on hyvä olla mittauspisteen tai siitä vastaavan tietolähteen nimi, koska sen voidaan olettaa olevan tarpeellisin tieto ja käytetyin rajausehto. Mittauspistetiedon avulla olisi esimerkiksi mahdollista erotella toisistaan esimerkkikuvan ilman ja kiitotien lämpötilat.

Minkä tahansa tietopuun tiedon kannattaa sallia olevan rajaavana tekijänä tietoa tilattaessa, jotta tilauksia voidaan myöhemmin rajata myös muilla kuin nyt nähtävissä olevilla rajaustarpeilla. Tällöin tulevaisuudessa mahdollisesti ilmenevät rajaustarpeet eivät vaadi muutoksia datapalvelimeen. Näitä rajauksia voidaan tehdä yksinkertaisesti: metatiedon perusteella tehdään esikarsinta, jonka jälkeen eteenpäin lähetetään vain ne tietopuut, joita lisärajaukset varsinaisen tiedon suhteen eivät karsi pois.

Varsinaisiin tietoihin liittyy myös milloin tietopaketti on julkaistu tai minkä lähetyksen osa se on. Näitä tietoja ei kuitenkaan kirjoita puuhun tietoa julkaiseva asiakasmoduuli, vaan tiedon vastaanottava datapalvelimen yksikkö. Tiedot ovat tarpeen viimeisimmän julkaistun tiedon löytämiseksi ja välistä pudonneiden lähetysten tunnistamiseksi. Esimerkiksi jonkin datapalvelimen yksikön saadessa kahdelta muulta yksiköltä niiden asiakasmoduulien tuottamana samaa tietoa, voidaan eteenpäin toimittaa vain uudempi näistä kahdesta tiedosta. Myös tietyn aikavälin aikana julkaistujen tietojen tilaaminen tulee samalla yksinkertaiseksi.



Kuva 5.1: Esimerkki varsinaisten tietojen suhteesta niitä kuvaaviin metatietoihin

Toisaalta lähetysajan kirjaaminen luo haasteita toiminnalle. Esimerkiksi viiden viimeisimmän tietyn tahon julkaiseman tiedon löytäminen ei kellonajoilla ole yksiselitteistä, koska tiedon julkaisukertojen välillä kulunut aika ei ole vakio (vertaamalla kahden lähetyksen kellonaikaa ei voi havaita niiden välistä pudonnutta lähetystä). Lisäksi lähetyksen vastaanottavan datapalvelimen yksikön olisi luotettava tiedon kirjanneen yksikön kellonajan olevan ainakin suunnilleen oikein: jos eri yksiköiden kellot ovat eri ajassa, voivat datapalvelimen yksiköt tehdä virheellisiä päätelmiä tuoreimmasta tiedosta, kun samaa tietoa saa useammalta lähteeltä.

Tältä synkronointiongelmalta voidaan välttyä käyttämällä asiakaskohtaisia julkaisunumeroita, eli numeroimalla jokainen asiakasmoduulilta vastaanotettu lähetys. Näin tällaista synkronointiongelmaa ei tulisi eteen, sillä asiakasmoduulin ei oleteta vaihtavan toiselle datapalvelimen yksikölle. Mikäli asiakasmoduulin kuitenkin sallitaan vaihtaa käytettävää datapalvelimen yksikköä, tulee vastaava synkronointiongelma eteen myös lähetysnumeroiden kanssa – eri yksiköiden pitää synkronoida lähetysnumerot niin, että ne eivät mene päällekkäin.

5.2. Metatiedot

Metatietojen tarkoitus on kuvata varsinaisiin tietoihin liittyviä asioita. Tällainen tietoon liittyvä tieto on esimerkiksi mitä mittayksikköä on käytetty kuvaamaan nimetyn tiedon tiettyä lehteä. Näin voidaan erotella esimerkiksi lämpötilan lähettäminen Fahrenheit- ja Celsius-asteina. Vaikka tiedon nimi onkin periaatteessa metatietoa, on jokaisen tiedon tunniste kuitenkin katsottu myös varsinaiseen tietoon kuuluvaksi osaksi. Näin sitä voidaan käyttää tunnisteena sitomaan metatiedot varsinaiseen tietoon niissä tilanteissa, joissa metatietorakenteeseen ei ole viitattu suoraan. Varsinaisen tiedon ja metatiedon eroja selventää graafinen esimerkki (kuva 5.1).

Metatietopuuta ei tarvitse luoda jokaiselle tietopuulle. Tietopuun sisältämän puun muodostaessa oman kokonaisuutensa on kuitenkin suotavaa luoda sille oma metatietopuu – isommassa puussa voidaan kertoa viittaus tähän puuhun redundantin tiedon tallentamisen välttämiseksi. Näin samaa kuvausta ja rakennetta voidaan hyödyntää useassa kohteessa: esimerkkikuvan tietojen alipuuta ”Tuuli” voidaan kenties hyödyntää jossain toisessa yhteydessä, mikäli sille luodaan oma metatietopuu. Aina tällaista rakennetta ei kuitenkaan ole järkevää luoda: siitä on hyötyä vain, mikäli jokin asiakasmoduuleista saattaisi haluta tilata vain tuota erillistä tietoa. Myös kyseisen tiedon lähetysten toistuvuus tulee huomioida – mikäli tiedosta tulee vain yksi lähetys, ei erillisen puun tekeminen ole niin kannattavaa kuin kerrottaessa tuon osan tietoja usein.

Kuten puiden erot esittelevästä kuvasta näkyy, metatietojen puu sisältää enemmän tietoja kuin niitä on siihen liittyvässä tietojen puussa. Jokaiselle varsinaisten tietojen puun lehdelle kuuluu vähintään yksikön kertova lehti metatietojen puussa. Lisäksi metatietoina on hyödyllistä kertoa muun muassa varsinaisten tietojen puun koko (lehtien määränä), luottamuksellisuusaste ja prioriteetti (jotka on ohjelmistoperheessä kuvattu yhteisenä numerokoodina), sekä mihin isompaan kokonaisuuteen puu mahdollisesti liittyy (jos tieto ei ole sidoksissa ylemmän tason kokonaisuuteen, jää kenttä tyhjäksi).

Julkaisijan tulisi aina kertoa mahdollisimman useat metatietojen puun lehdet – erityisesti luottamuksellisuusasteen metatieto on tärkeä, sillä datapalvelin ratkaisee tiedon käsittelytavat pääasiassa sen sisältämän tiedon perusteella. Yksittäisen puun luottamuksellisuusasteen tulisi olla vähintään yhtä luottamuksellisella tasolla kuin sen sisältämien tietojen luottamuksellisin taso on. Koko puu voi myös olla luokiteltuna luottamuksellisemmaksi kuin yksikään sen lehdistä, samalla periaatteellisella syyllä kuin on esitetty Valtiovarainministeriön esimerkissä [24, kohdan 5.8. alakohta (3)]. On

myös huomioitava, että kokoelma jonkin tason tietoja tulee luokitella tuota tasoa korkeammalle, jos tietoja säilytetään huomattavia määriä.

Metatietoja voidaan myös käyttää etsittäessä haluttuja tietoja. Näin esimerkiksi asiakasmoduuli voi ilmoittaa olevansa kiinnostunut Tampereen tiedoista. Tällöin asiakasmoduulin voidaan katsoa tilaavan kaikki sellaiset tiedot, joiden kuvauksesta löytyy viittaus Tampereeseen. Esimerkkitilanteen laskeutumistiedot olisivat näiden tietojen joukossa, sillä Tampere-Pirkkala sisältää viittauksen Tampereeseen.

Tätä ei kannata rajoittaa pelkkään mittauspaikkaan, vaan haku kannattaa suorittaa kaikkiin metatietojen alaisiin kohtiin. Näin mahdollistetaan esimerkiksi kaikkien lentokoneiden sijainti-, korkeus-, suunta- ja nopeustietojen merkitseminen haettavaksi, ja lisäämällä sijaintiin liittyvän rajoituksen voidaan näillä tiedoilla valvoa lentoliikennettä jollain tietyllä alueella niin, ettei tietyn korkeuden yläpuolisia lentokoneita huomioida – tarpeellinen rajaus esimerkiksi lähilennonjohdon ja alueellisen lennonjohdon vastuualueiden erottamiseksi.

5.3. Tietojen siirron edellytykset

Tilauksen yhteydessä datapalvelin päättelee tilattavan tietopuun vastaanottamansa metatiedon perusteella. Tilaajan vastuulla on tämän vuoksi kuvata sen tarvitsema tieto mahdollisimman hyvin, jotta siihen tarvittavat puut voidaan muodostaa tehokkaasti. Julkaisijan vastuuna on vastaavasti muodostaa tuottamansa tieto hyvin muotoilluksi puuksi, ja kuvata se tarpeellisella määrällä metatietoa. Julkaisijalle on jätetty täysi vapaus päättää kuvauksen puurakenteesta: datapalvelimen ei siis tarvitse varautua muuttamaan julkaistujen tietojen puurakennetta.

Listaus 5.1 on muodostettu esimerkiksi eräästä hyvin muotoillusta puusta. Listauksessa kuvattu puumainen lämpötilojen esittäminen tukisi verkon tehokasta hyötykäyttöä enemmän kuin edellä esitellyn (kuva 5.1) kaltainen rakenne. Metatietoina molemmilla uusilla lämpötilan lehdillä voisi olla esimerkiksi {”Mittauspiste”, ”Celsius”} normaalien puuta kuvaavien kenttien lisäksi. Se on kokonaisuutena selkeämpi, koska lämpötilat on selkeästi eritelty eri paikoista mitattavaksi. Siinä on paremman rakenteen avulla myös helpotettu uusien tietojen lisäämistä. Esimerkkinä tästä siihen on lisätty kiitotien liukkausaste, jolle ei ole luonnollista paikkaa aiemmin esitellyn kaltaisessa rakenteessa.

Datapalvelin voi korkeintaan yhdistää useita tietopuita yhdeksi tilatuksi kokonaisuudeksi. Esimerkiksi jos datapalvelimen yksiköltä on tilattu jotain tietoa pidemmillä aikaväleillä kuin asiakasmoduuli sitä toimittaa, voi datapalvelin myös yhdistää kaikki aikavälillä saapuneet tiedot yhdeksi lähetykseksi, tai jos jokin lähetys kasvaa käytettävissä olevaan siirtokapasiteettiin nähden suureksi, voi datapalvelin jakaa jonkin lähetyksen osiksi. Puun rakenteeseen ei tule koskea, vaikka tällaista jakoa tai yhdistämistä tapahtuu. Esimerkkitapauksessa julkaiseva taho olisi voinut koota lämpötilat omaksi puukseen, mutta datapalvelin ei tähän kykene – sen tehtävä on siirtää tietoa eikä yrittää suorittaa syvällistä analyysiä sen rakenteesta.

Sekä paloittelu että yhdistäminen ovat tarpeen sen varmistamiseksi, ettei reitityskerros ajaudu varaamaan reittiä pitkäksi aikaa jonkin tiedon lähettämiseksi, jolloin muu tärkeämmän prioriteetin tieto voi joutua odottamaan alhaisemman prioriteetin toimia. Osittain tehtävän kykenisi hoitamaan myös reitityskerros, mutta hoitamalla samaa tehtävää molemmilla tasoilla saadaan toimintaa säädettyä hienojakoisemmin. Tämä näkyy parhaiten esimerkiksi tärkeän prioriteetin videon lähettämisen yhteydessä.

```
<Tiedot MetaID = '…' Tyyppi = 'Laskeutumisen säätiedot'>
        <Aikaleima> 2004-12-31 23:58:02 </Aikaleima>
        <Paikka>
                <Nimi> Tampere-Pirkkalan Lentokenttä </Nimi>
        </Paikka>
        <Säätiedot>
                <Paikka>
                        <Nimi> Torni </Nimi>
                </Paikka>
                <Näkyvyys> 500 </Näkyvyys>
                <Tuulitiedot>
                        <Suunta> NW </Suunta>
                        <Nopeus> 3.1 </Nopeus>
                </Tuulitiedot>
                <Ulkolämpötila> 15 </Ulkolämpötila>
        </Säätiedot>
        <Säätiedot>
                <Paikka>
                        <Nimi> Kiitotie 24/06 </Nimi>
                </Paikka>
                <Lämpötila> 21 </Lämpötila>
                <Liukkaus> Kuiva </Liukkaus>
        </Säätiedot>
</Tiedot>
```

Listaus 5.1: Säätietojen tietopuun laajennettu kuvaaminen XML-kielen avulla

Jokaista sen bittiä ei tarvita, mutta koko video on kuitenkin tärkeällä prioriteetilla. Jos jakoa suoritetaan vain reitityspalvelussa, joutuisi se mahdollisesti jättämään muita lähetyksiä välistä, koska se näkee vain paketteja tärkeällä prioriteetilla. Datapalvelimen tasolla voidaan taas nähdä että tässä menee tärkeän prioriteetin videota, mutta reitille on menossa myös muutama lyhyt ja aika tärkeä viesti – tällöin datapalvelimen tasolla voidaan paremman kokonaiskuvan ansiosta hidastaa videon siirtoa hiukan, jotta nämä lyhyet viestit saadaan kulkemaan ajallaan.

Koska aina ei voida luotettavasti ennustaa johonkin tietoon tulevaisuudessa liitettäviä lisätietoja, tulee mahdollistaa puiden rakenteen muokkaaminen. Esimerkiksi

kiitotielle ei välttämättä ole ollut tarvetta ilmoittaa liukkautta ennen pidempien kiitoteiden tarvetta, jos niiden pituus on ollut riittävä kaikissa keleissä. Jotta kaikkia tietoja vastaanottavia tahoja ei tarvitse muuttaa rakenteen muutoksen vuoksi, hyödynnetään kohdassa 3.4.4 mainittua tulkkimoduulia. Se muokkaa uudemmasta rakenteesta säännöllisten lausekkeiden avulla vanhemman kaltaisen rakenteen. Tulkkimoduulin käyttäminen vaatii kuitenkin sen, että tilauksen yhteydessä mainitaan tiedosta sen haluttu versio. Näin säästytään muutoksen kertautumiselta ja luodaan mahdollisuus siirtää myös tulevaisuudessa esiteltäviä rakenteita.

5.4. Siirrettävän tietomäärän vähentäminen

Yksittäiseen julkaisuun liittyy yleensä monta lähetystä. Tästä johtuen varsinaista tietoa siirtävän puun ei enää siirtovaiheessa kannata sisältää metatietoja. Ne kannattaa jakaa erikseen muille datapalvelimille esimerkiksi julkaisu- tai tilaustapahtuman yhteydessä. Näin sitä ei tarvitse välittää moneen kertaan jokaisen datapaketin mukana. Toisaalta tämä lisää muistin tarvetta, koska jokainen datapalvelimen yksikkö tulee tietämään suurimman osan metatiedoista – vaikka ne eivät jotain tietoa välttämättä tarvitsisi.

Paitsi tilaukseen liittyvien tietojen siirron suhteen, myös datapalvelinten yksiköiden välisessä tilauksien ja julkaisujen vaihdossa pyritään mahdollisimman vähäiseen liikenteeseen. Tätä kohti voidaan edetä esimerkiksi tarkastussummien avulla kertomalla aluksi kaksi eri tarkastussummaa: toinen lasketaan kaikkien julkaisujen nimistä ja toinen vastaavasti tilauksien nimistä. Molemmissa tapauksissa tarkastussummaan lasketaan myös niihin liittyvien metatietojen puiden tiedot. Tilauksien yhteydessä mukaan täytyy lisäksi laskea jo kerrottujen tietolähetysten tunnisteet, jotta voidaan havaita muutos jo lähetettyjen tietojen kohdalla. Kyseessä on siis kaksi eri tavalla laskettua, erillistä tarkastussummaa, joiden avulla voidaan vertailla ovatko J&T-rekisterin tiedot edelleen samanlaisia kuin ennenkin.

Käytettäessä tarkastussummia on kuitenkin huomioitava mahdollisuus yhteentörmäyksiin eli tilanteisiin, joissa tiedot ovat muuttuneet, mutta tarkastussumma pysyy samana. Tähän voidaan puuttua kahdella tapaa. Ensimmäinen vaihtoehto on tehdä ylimääräinen tarkastus vertaamalla tiedon lisäämisen jälkeen saatavaa tarkastussummaa edelliseen, jolloin voidaan havaita yhteentörmäys ja tarvittaessa kasvattaa tarkastussumman pituutta – tai merkitä tieto törmäyksestä talteen, jotta tarkastussumman lähetys jätetään väliin. Toinen vaihtoehto on käyttää varmempia, kryptografisia tapoja laskea tarkastussumma. Ne ovat varmempia, koska jokainen bitti vaikuttaa suurempaan osaan tarkastussumman biteistä, jolloin ne tuottavat kaikista pienistäkin muutoksista hyvin erilaisia tarkastussummia – myös yhteentörmäysten todennäköisyys laskee samalla [8]. Näistä jälkimmäisen heikkoutena on huomattavasti suurempi laskentarasitus verrattuna tavallisiin hajautusfunktioihin, joita tehtävään yleensä käytetään.

Ensimmäisessä menetelmässä tietoja lähettävä datapalvelimen yksikkö voi pyrkiä havaitsemaan tilanteen tiedon muutoksen yhteydessä tarkastamalla tarkastussumman sekä ennen tiedon muuttamista että sen jälkeen. Jos tarkastussumma on säilynyt samana

muutoksen yhteydessä, voi yksikkö lisätä ylimääräisen numeron tarkastussumman perään. Tämä ylimääräinen numero taas erottuu vastaanottajalle vertaamalla tarkastussumman pituutta sen oletettuun pituuteen. Numeron esiintyessä voi tietoa vastaanottava yksikkö tarkastaa onko silläkin sama numero tallessa vai ei, ja tehdä näin päätelmän tietojen mahdollisesta muuttumisesta.

Vaikka näin toimien voidaan ratkaista ongelma täysin, tulee siitä kuitenkin verkkoon lisää siirrettävää tietoa ylimääräisten numeroiden kohdalla. Paras ratkaisu on siis käyttää molempia menetelmiä, koska datapalvelimen yksiköiden oletettiin olevan resursseiltaan runsaita. Näin ylimääräistä numeroa joudutaan lähettämään hyvin harvoin, koska kryptografisia menetelmiä käytettäessä törmäysten esiintyminen on erittäin harvinaista samankaltaisten tietojen yhteydessä.

Mikäli tiedot vastaanottava datapalvelimen yksikkö saa toisesta (tai molemmista) tarkastussummaksi eri tuloksen, voi se seuraavaksi pyytää tietoa lähettävältä yksiköltä puiden nimet ja kunkin puun oman tarkastussumman. Tämän jälkeen voidaan jatkaa etsimällä muuttuneet puut ja tilaamalla vain näiden yksittäisten muuttuneiden puiden kuvaukset. Koska tilaukset ja julkaisut eivät kokonaisuutena yleensä muutu synkronointien välissä, voidaan tällä tavalla vähentää datan siirtotarvetta huomattavasti. Tarkastussummien lähettäminen kannattaa kuitenkin ohittaa niissä tilanteissa, joissa niiden siirtäminen tarvitsisi enemmän siirtokaistaa kuin varsinaisen puun kertomiseen menisi.

Tilaustietoihin kuuluu muun muassa mitä julkista avainta tilaaja haluaa käytettävän tiedon salaamiseen ja kuinka hyvin tilaus on jo tähän mennessä täytetty. Periaatteessa tilauksen mukana voitaisiin kuljettaa pelkästään tietoa ajankohdasta tai lähetysnumerosta, jota uudempia viestejä halutaan. Tämä voi kuitenkin aiheuttaa samojen tietolähetyksien moninkertaista lähettämistä, sillä muut datapalvelimen yksiköt eivät tietäisi, onko jokin rajaa tuoreempi tieto lähetetty kohteelle vai ei.

Koska mikään datapalvelimen yksiköistä ei omaa täydellistä tietoa kaikista lähetyksistä, voitaisiin tämä raja myös asettaa väärään kohtaan – jokin datapalvelimen yksiköistä voi välimuistinsa sisällön vuoksi luulla, että lähetys 100 on vanhin lähetetty, kun jokin toinen datapalvelimen yksiköistä voi tietää vielä vanhempia. Kertomalla jo toimitetut osat voi jokainen tilauksen vastaanottava datapalvelin tarkastaa mahdollisuutensa toimittaa siitä puuttuvia osia, ja lähetyksiä siirretään uudestaan mahdollisimman vähän. Asiakasmoduulilta tilauksen vastaanottanut datapalvelimen yksikön tulee kirjata hallussaan olevat osat jo toimitetuiksi, jotta muilta yksiköiltä ei tilata turhia tietolähetyksiä. Koska tieto tilauksesta pyritään lähettämään aina julkaisijoille asti, tulee matkan varrelta myös uusimmat tiedot lähetettyä tilaajalle.

5.5. Uudelleenlähetysten välttäminen

Toistuvien lähetysten täydellinen eliminointi vaatisi jokaisen datapalvelimen yksikön tietävän koko ajan muiden datapalvelimen yksiköiden teot. Listaamalla jo lähetetyt lähetykset voidaan pyrkiä poistamaan toistuvat lähetykset mahdollisimman aikaisessa vaiheessa. Lähettämällä tilaus aina julkaisijalle asti voi tietoa julkaiseva taho valita

lähetettävät paketit. Toisaalta tilauksen kulkeman reitin varrella jo valmiiksi olevat tiedot kannattaa lähettää kohti tilaajaa.

Kirjaamalla ne tilaustietoihin jo lähetetyiksi voidaan estää julkaisevaa tahoa lähettämästä kyseisiä tietoja uudestaan. Jos tilaajaa kohti kuitenkin on tulossa jo lähetetty tieto, voivat reitin varrella olevat datapalvelimen yksiköt tiputtaa sen pois, mikäli ne löytävät kyseisen lähetyksen jo lähetettyjen joukosta. Näin tilaukselle toimitettujen lähetysten listan avulla vähennetään verkon liikennettä.

Uudelleenlähetysten havaitsemista helpottavana tunnisteena voidaan käyttää esimerkiksi julkaisun tietolähetyksen yksilöivää tunnistetta tai tietorakenteen yksittäisen tietopaketin julkaisutapahtuman kellonaikaa riittävällä tarkkuudella. Näistä lähetystunnisteet ovat suositeltavampi vaihtoehto, sillä näin vältetään ongelma kellonaikojen synkronoinnista ja sellaisen tarkkuuden määrityksestä, jolla kaikki mahdolliset julkaisutapahtumat asettuisivat eri ajanhetkelle (jolloin tietyn asiakkaan tiedon julkaisuajankohta on aina yksilöllinen ja jota voisi siis käyttää tunnisteena). Se ei kuitenkaan ole ilman ongelmia: jos asiakasmoduuli menettää yhteyden datapalvelimen yksikköön, ja tämän seurauksena yhdistetään johonkin toiseen yksikköön, ei aina olisi varmaa tietoa oikeasta lähetysnumerosta saatavilla.

Erityisen vaikean tilanteesta voi tehdä yhteyden katkeaminen tietoa lähetettäessä, sillä silloin ei tiedetä ehtikö lähetys saapua perille (jolloin lähetysnumeroa pitäisi kasvattaa) vai siirtyikö se vain osittain (jolloin se pitäisi pitää samana). Tätä ongelmaa voi vähentää lähetyksen onnistumisen varmistamisella: jos varmistusta ei saada toimitettua, katsotaan se epäonnistuneeksi sekä asiakasmoduulin että datapalvelimen toimesta. Tämä tosin siirtää ongelman varmistuksen vastaanottamiseen. Varsinaisiin tietoihin verrattuna sen koko on kuitenkin huomattavasti pienempi, joten yhteyden katkeamisen tapahtuminen sen aikana on vastaavasti todennäköisyydeltään pienempi. Tämä ei poista ongelmaa kokonaan, mutta vähentää sen vaikutuksia huomattavasti. Pääasiassa datapalvelimen tarvitsee huomioida tämä ongelma vain asiakkaiden ja datapalvelimen yksiköiden välisen liikenteen yhteydessä, sillä välitettäessä tietoa eri datapalvelimen yksiköiden välillä tämä ongelma on reitityskerroksen vastuulla.

5.6. Varsinainen siirtäminen

Puut siirretään tuoteperheen omilla tietorakenteilla. Niitä ei käydä lävitse tässä työssä. Kun tietoja siirretään jonkin siirtotien ylitse, on hyödyllistä aina sitoa tietoon siihen liittyvän datan määrä eli lähetyksen koko. Tietoa voidaan siirtää myös asiakkaalta toiselle ilman verkon yli tapahtuvaa liikennettä, mikäli molemmat asioinnin osapuolet ovat liittyneet saman datapalvelimen yksikön alaiseksi ja ovat siihen suoraan liitoksissa – käytännössä siis saman koneen eri prosesseja. Asiakkaan puolelta tämän ei pitäisi erota verkon yli tapahtuvasta tietojen siirrosta.

Tietojen siirto reitityskerroksessa tapahtuu puskureiden kautta. Tiedon tyypin ja julkaisuilmoituksen perusteella sille voidaan valita eteenpäin lähetystä varten joko putkimainen toimintatapa (puskurin koko on yksi, ja siinä on aina joko viimeisin

julkaistu tieto tai ensimmäinen lähettämätön tieto – muut tiedot poistuvat) tai dynaamisen reitin toiminta (jolla pyritään minimoimaan katoavat tiedot, toisin sanoen tätä käytetään kun tietoja ei saa hukkua).

Esimerkiksi putkimaisella toimintatavalla voidaan jatkuvasti päivittyvästä paikkatiedosta pitää vain viimeisin tieto voimassa, ja käyttää dynaamisen reitin toimintaa viestien välittämiseen niiden toimituksen varmistamiseksi. Asiakasmoduulin on kuitenkin päätettävä käytettävän reitin tyyppi: jos halutaan saada talteen tahon käyttämä kulkureitti, tulee käyttää dynaamista reittiä myös paikkatiedon välittämiseen. Vastuu toimintatavan valinnasta on asiakasmoduulilla, koska datapalvelimella ei ole mahdollisuuksia tehdä luotettavaa analyysiä tiedon käyttötarkoituksesta.

Paitsi tietojen siirtoon datapalvelimen yksiköiden välillä, myös jokaiselle asiakkaalle kannattaa luoda omat puskurinsa. Luomalla omansa sekä tietojen lähetykseen että vastaanottoon voidaan varmistaa tietojen siirtymistä sekä datapalvelimelle että datapalvelimelta – muuten laskentaintensiivinen hetki voi estää lähetyksen vastaanottamisen, kun saapuvia ei ehditä käsittelemään ajoissa. Tällaisia puskureita kannattaa luoda erityisesti reitittävissä solmuissa sijaitseville asiakasmoduuleille.

Tätä varten datapalvelin voi pyytää verkkokerrokselta yhteydet näiden puskureiden tietojen välittämiseen, mikäli yhteyden toinen osapuoli on jossakin toisessa solmussa. Tällöin puskurit luodaan reitityskerroksen käyttöä varten, mutta niiden kohteeksi asetetaan nämä asiakasmoduulit. Reitityskerrokselle voi siis tulla useita puskureita samalle asiakkaalle. Datapalvelimen tehtäväksi jää siis sitoa kohde oikein, mutta oikean toimitusosoitteen varmistuksen tehtävä siirtyy sen paremmin hoitamaan kykenevälle reitityskerrokselle.

Tietojen välittämiseen voitaisiin hyödyntää myös muutostietoja. Esimerkiksi jatkuvasti muuttuvan paikkatiedon välittämiseen kerrottaisiin aluksi lähtöpaikka ja sen jälkeen muutostietoa. Tulkkimoduuli voisi muuttaa tämän tavalliseksi koordinaattiesitykseksi, mikäli asiakkaan tilaama tiedon muoto näin edellyttää. Yksinkertaisen paikkatiedon tapauksessa tällä toimintatavalla ei todennäköisesti saavutettaisi merkittävää etua, mutta monimutkaisemman tiedon yhteydessä etu voi olla merkittävä.

Datapalvelin voi kuitenkin hyödyntää tällaista menettelyä omassa toiminnassaan, esimerkiksi välittämällä tilauksien täyttöasteen muutoksia eikä aina tilausta kokonaan. Näin muutosten yhteydessä voidaan säästää verkon käyttöä. Ennustettu verkon toimintavarmuus on kuitenkin heikko, ja yksittäiset lähetykset voivat viivästyä ennalta tuntemattoman ajan. Tämän vuoksi toimintaa tietojen välittämisen suhteen ei suositella datapalvelimen ensimmäisiin versioihin. Se on kuitenkin potentiaalinen jatkokehitysmahdollisuus, ja tekniikkaa voidaan mahdollisesti hyödyntää myös tilausten ja julkaisujen muutosilmoituksia lähettämällä.

6. DATAPALVELIMEN YKSIKÖN RAKENNE

Itse datapalvelimen yksikön rakenne voidaan jakaa viiteen itsenäiseen osaan, joista jokaisella on omat vastuualueensa. Kukin näistä osista esitellään tämän luvun erillisissä kohdissa, joissa kerrotaan myös kyseisen rakennemoduulin toiminnasta yleisellä tasolla. Lopuksi viimeisessä kohdassa kerrotaan näiden osien yhteistoimintatapa, jonka avulla ne muodostavat datapalvelimen yksikön.

6.1. Semantikko

Semantikon tehtävänä on nimensä mukaisesti hoitaa semantiikan ymmärtäminen, eli se purkaa tilaajan pyytämien käsitteiden sisällöt ja määrittää niiden muodostaman kokonaisuuden avulla, mikä tai mitkä yläkäsitteistä on tilattava. Se voi esimerkiksi metatietojen perusteella käsittää tilauksen osan ”lähitapahtumat” tarkoittavan samalla lentokäytävällä olevia, 10km etäisyydelle saapuvia lentokoneita (tilaaja on kertonut olevansa lentokone) tai tälle lentokentälle tulossa olevat koneet ja niiden arvioitu saapumisaika (tilaaja kertoo olevansa laskeutumiset listaava moduuli).

Molemmat esimerkin tapaukset voisivat hyödyntää sellaista lentodata-tietopuuta, jonka avulla välitettäisiin lentokoneen lentämiseen liittyviä tietoja. Mikäli lentokäytävä johtaisi samalle kentälle, myös varsinaiset tiedot olisivat samoja. Vaikka varsinainen tilaus olisi näillä tapauksilla erilainen, voidaan koko lentodatan tilaamisella saavuttaa etua kokonaisuutta tarkasteltaessa – esimerkiksi jos nämä kaksi tilaajaa esiintyisivät erillisillä datapalvelimen yksiköillä ja tieto kulkisi samalla reitillä. Koska tietoa odotetaan toimitettavaksi useille yksiköille, tämä oletetaan yleisimmäksi tapaukseksi.

Semantikko on datapalvelimen yksikön monimutkaisin osa. Tämä johtuu sille osoitetusta päättelyvastuusta: on löydettävä tilauksen täyttämisen kannalta tehokkain tapa valita yksi tai useampia tietopuita. Täten semantikko tulee suurilta osin ratkaisemaan datapalvelimen toiminnan lopullisen tehokkuuden: mikäli semantikko ei kykenisi päättelemään tarvitsevansa kahta pienempää puuta jo tilatusta isommasta puusta, laskisi verkon tehokkuus oleellisesti. Semantikon tulee siis ottaa huomioon myös muut, jo olemassa olevat tilaukset: jos edellä mainitut kaksi alipuuta tilataan erikseen, ei se ole yhtä kannattavaa kuin yhden kokonaisen puun tilaaminen – jolloin muutkin datapalvelimen yksiköt kykenevät paremmin hyödyntämään tuota tietoa.

Käytännössä tarvetta semantikolle ilmenee kerran jokaisen tehdyn tilauksen yhteydessä. Tässä yhteydessä sen kannattaa tarkastaa, onko joitain vanhoja tilauksia mahdollista yhdistää. Kun tilaus on kerran muodostettu, voidaan sitä hyödyntää suoraan, eikä semantikon palveluita tarvita. Tilaukset ovat yleensä myös pitkäikäisiä,

joten semantikon palveluksia ei tarvita kuin hyvin harvoin. Näistä syistä johtuen semantikkoa ei kannata sijoittaa omaksi säikeekseen.

6.2. Liittäjä

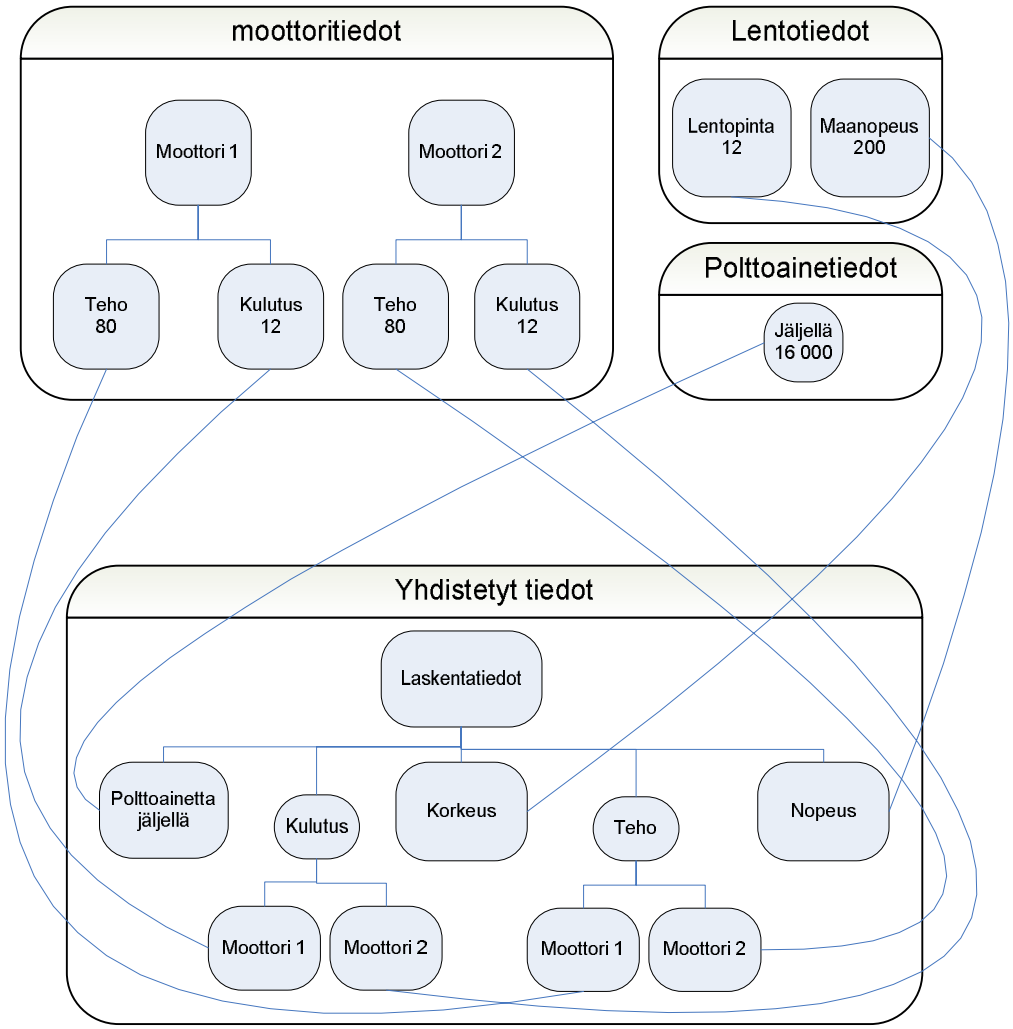
Liittäjän nimi kuvaa hyvin sen tehtävää: kun datapalvelin saa tilattuja tietoja, on sen liitettävä mahdollisesti useasta lähteestä saapuneet tiedot varsinaisessa tilauksessa ilmoitettuun muotoon. Datapalvelimen yksikkö on esimerkiksi voinut tilata lentokoneen moottoritiedot (tämän hetkinen käyttöteho, kulutus), lentotiedot (korkeus liitämisen huomioimista varten) ja polttoainetiedot (säiliöissä jäljellä oleva määrä) voidakseen laskea jäljellä olevan maksimikantaman nykyisellä kulutuksella.

Toiminnan mahdollistamiseksi liittäjän tulee voida yhdistellä tietopuita tarvittaessa sekä ennen että jälkeen ulkopuolisen tulkkimoduulin (kohta 3.4.4) kutsumista. Tälle tulkkimoduulille kerrotaan tilauksen metatiedot sekä tilauksen tietopuut ja niiden muodot. Tulkki hoitaa tarvittavat yksikkökonversiot ja karsii tilaukseen liittymättömät tiedot pois. Tämän esimerkin tapauksessa siis vain arvio jäljellä olevasta matkasta välitettäisiin liittäjälle takaisin. Liittäjä ohjaa palautetut tiedot takaisin käsittelijälle ja sieltä ne ohjautuvat asiakasrajapinnan kautta ne tilanneelle moduulille.

Varsinainen tietojen yhteen liittäminen voi tapahtua esimerkiksi luomalla puu, jossa kerrotaan linkit varsinaisten tietojen muistiosoitteeseen. Tällainen tapa on suorituskyvyltään tehokkaampi sekä muistinkulutukseltaan kevyempi verrattaessa varsinaisten tietojen kopiointiin uuteen puurakenteeseen. Samalla myös estetään ohjelmiston väliaikaisen muistinkulutuksen hallitsematon kasvu, ja tarjotaan hyvä tallennuspaikka tiedoille, jos joudutaan odottamaan myöhemmässä lähetyksessä toimitettavia muita tietoja.

Tämä tulkille välitettävä muoto on esitetty yksinkertaistetussa muodossa graafisesti (kuva 6.1). Kaarevilla viivoilla kuvataan siinä muistiviittauksia. Varsinaisissa puissa olisi todennäköisesti myös muita tietoja kuin vain kuvassa esitetyt, mutta nämä on jätetty pois kuvasta selkeyden parantamiseksi. Samasta syystä siinä ei myöskään ole esitetty tietoihin liitettäviä metatietoja.

Koska liittäjä joutuu yhdistämään tietoja ja etsimään yhteyksiä eri alkioiden ja datarakenteiden välillä, tulee sen toiminta olemaan hyvin laskentaintensiivistä. Aina tilauksen tietoja vastaanotettaessa tarvitaan liittäjän toimia, mutta monimutkaisen tilauksen käsittely ei kuitenkaan saisi hidastaa koko datapalvelimen yksikön toimintaa. Tämä ongelma voidaan ratkaista tekemällä tästä osasta oma säikeensä. Näin käsittelijämoduuli kykenee samanaikaisesti ajamaan esimerkiksi tietoturvatarkastuksia seuraavan tiedon käsittelyn nopeuttamiseksi.



Kuva 6.1: Tietojen liittäminen useasta puusta yhdeksi puuksi

6.3. Tietoturvaaja

Tietoturvaajan tehtävänä on vastata turvallisuuteen liittyvistä palveluista. Suurin vastuu sillä on päätellä tiedon sisällön perusteella sen turvaamisen vaatimista toimenpiteistä (toimintamallia kutsutaan kirjallisuudessa muun muassa lyhenteellä CBIS eli Content Based Information Security). Käytännössä tämä tulee pohjautumaan tietojen julkaisijan ilmoittamaan luottamuksellisuusasteeseen, jonka pohjalta voidaan tehdä päätelmä sen riittävän turvalliseen välittämiseen tarvittavista toimenpiteistä. Myös luokittelemattomia tietoja voidaan sallia käsiteltävän, mutta silloin näiden tietojen osalta ei voida olla varma riittävästä tietoturvasta.

Mikäli datapalvelimelta pyydetään tietoja, joihin niiden pyytäjällä ei ole oikeutta, tai tiedoille yritetään tehdä muu kielletty operaatio (esimerkiksi välittää salassa pidettävää tietoa eteenpäin ilman salausta), tulee tietoturvaajan pysäyttää tietojen käsittely. Toimenpiteen suoritus tulee voida pysäyttää myös silloin, kun huomataan datapalvelimen yksikön käsittelyyn saaman tiedon olevan sen oikeuksien tai mahdollisuuksien ulkopuolella, tai kun tietoihin liitettyjä vaatimuksia huomataan rikotun muulla tavoin. Näin voidaan varautua esimerkiksi tilanteeseen, jossa olisi

vahingossa kytketty luokiteltua tietoa tuottava komponentti sellaiseen datapalvelimen yksikköön, jolla ei ole riittäviä oikeuksia tai mahdollisuuksia sen käsittelyyn.

Tämä tilanne tulee ilmoittaa datapalvelimen käyttöliittymän kautta sen operoijalle, ja vastaavasti käsittelijän kautta asiakasrajapintaan. Näin asiakaskomponentti voisi esimerkiksi kytkeä itsensä pois päältä tai antaa oman käyttöliittymänsä kautta hälytyksen tilanteesta. On suositeltavaa tehdä molemmat. Tieto kannattaa lähettää molemmille tahoille, sillä niitä ohjaamassa voivat olla eri henkilöt. Lisäksi henkilöt voivat sijaita huomattavan etäisyyden päässä toisistaan.

Tietoturvaaja vastaa myös AAA-mallin (kohta 2.2.1) noudattamisesta. Käytännössä tämä tarkoittaa sitä, että aluksi varmistetaan kuka vastapuoli on, sitten mitkä ovat sen oikeudet, ja lopuksi kirjataan tapahtunut palvelupyyntö lokitietoihin. Vastapuolen varmistukseen osallistuu myös reitityskerros sekä käsittelijä. Mikäli vastapuolta ei tunnisteta tarpeeksi luotettavasti, ei siltä edes yritetä tarkastaa oikeuksia. Tunnistuksen epäluotettavuuden takia hylkääminen tulee kuitenkin kirjata lokitietoihin.

Jos tietoturvaaja huomaa, ettei datapalvelimen yksikkö kykene käsittelemään tietoa sen luottamuksellisuusasteen vaatimusten mukaan, tulee sen voida välittömästi poistaa nämä tiedot käsittelystään. Näihin mahdollisuuksiin vaikuttavat muun muassa datapalvelimelle sen käynnistyksen yhteydessä luettujen asetusten määrittämät salausprotokollat. Esimerkiksi jos tieto vaatii AES salauksen 512-bittisellä avaimella, ja datapalvelimen konfiguraatio ei tarjoa tätä (vaan vain heikompia salauksia tai tasoja kuten AES 256-bittisellä avaimella), poistetaan tiedot muistista ja ilmoitetaan tilanteesta asianomaisille. Tilanne tulkitaan ikään kuin datapalvelimelle ei olisi myönnetty oikeuksia tarkastella tämän tason tietoa: hylkäyspäätös syineen kirjataan lokipalveluun.

Koska tietoturvaaja hoitaa tietojen salauksen ja purkamisen salausmoduulin kautta, sen varsinainen oma toiminta keskittyy tietoihin liittyvien oikeuksien tarkastukseen. Tietoturvaa joudutaan kuitenkin tarkastamaan useilla eri tasoilla, mikä tekee sen toiminnasta laskentaintensiivistä. Lisäksi tietoturvaajan tulee estää muita datapalvelimen yksikön osia vaikuttamasta sen tekemiin päätöksiin. Tämän vuoksi se tulisi sijoittaa mahdollisimman erilliseksi komponentiksi tekemällä siitä esimerkiksi oma prosessinsa, jolloin sen toimintaan on vaikeampi vaikuttaa sitä suorittavan tietokoneen ulkopuolelta esimerkiksi aiheuttamalla ylivuotoja datapalvelimen yksikön muissa osissa, koska käyttöjärjestelmä pyrkii estämään eri prosessien väliset tahattomat vaikutukset. Vaikka tietoturvaajaan on tällöin helpompi vaikuttaa samalta koneelta (sen toimintatavat käyvät helposti ilmi seuraamalla yksittäistä komponenttia sen sijaan että pitäisi seurata kaikkia kokonaisuuden osia sekä sen muistialue on erillinen jolloin analysoitavana on vähemmän muistia), oletetaan datapalvelimen yksiköitä suorittavat tietokoneet suojatuiksi kohdan 4.3.4 mukaisesti haltuunottoa vastaan – jolloin voimme olettaa, ettei datapalvelimen yksikköä ajavalta tietokoneelta kohdistu datapalvelimen yksikköön uhkia. Samalla muu osa datapalvelimen yksikön toteutusta kykenee jatkamaan työskentelyä samalla kun tietoturvaaja suorittaa omia toimiaan, mistä saadaan suorituskykyetua useamman prosessorin omaavissa tietokoneissa.

6.4. Puskuroija

Puskuroijan tehtävänä on hallita puskureita. Nämä puskurit on tarkoitettu tasaamaan asiakasmoduulin ja datapalvelimen välistä viestintää, ja niitä avataan jokaiselle yksi lähetystä ja toinen vastaanottoa varten. Lisäksi puskureita voidaan avata eri asiakasmoduulien väleille. Datapalvelimen kussakin yksikössä tulee täten olemaan käytössä runsas määrä eri kohteisiin meneviä puskureita. Tällöin on todennäköistä, ettei käyttöjärjestelmä kykene yleisellä tasolla hallitsemaan puskureita yhtä tehokkaasti kuin erillinen toteutus pystyy, koska käyttöjärjestelmä joutuu toimimaan yleisemmällä tasolla kuin erikoistettu toteutus toimii.

Tällaisia puskureita voi olla kahta tyyppiä, joko putken tapaisia tai päästä päähän tyyppisiä. Putken tapaisissa puskureissa pidetään vain yksi tieto kerrallaan. Se on joko ensimmäinen tai viimeisin tieto, joka puskurin kautta halutaan välittää. Näin siinä voidaan pitää aina esimerkiksi viimeisintä paikkatietoa. Koska uusi tieto korvaa vanhan, on se aina ajan tasalla.

Vanhan tiedon automaattinen poistuminen ei kuitenkaan aina ole haluttu toimenpide. Esimerkiksi viestien yhteydessä ne kaikki halutaan välitettäväksi perille, eikä yhtäkään viestiä pitäisi hävitä. Tätä varten puskuroijan tulee tarjota mahdollisuus avata päästä päähän tyypin puskuri. Siihen mahtuu huomattavasti enemmän tietoja, ja korvaaminen voidaan asettaa tapahtumaan prioriteetin ja iän mukaisessa järjestyksessä. Näin estetään tärkeimpien ja uusimpien tietojen poistuminen puskurista, jos se pääsee täyttymään.

Ylärajat sekä puskureiden määrälle että koolle tulee asettaa datapalvelimen yksikön asetustiedoston kautta. Näiden arvojen muokkaus voidaan sallia myös ohjelman ajamisen aikana, jos sille katsotaan olevan tarvetta. Tällöin niiden pienentämisen yhteydessä ei kuitenkaan tule poistaa puskureissa jo olevia tietoja. Todennäköisempi käyttötapaus olisikin puskurien määrän tai koon ylärajan kasvattaminen, missä ei esiinny vastaavaa ongelmaa.

Puskuroijan tehtävänä on vastata puskureiden hallinnasta sekä niihin tulevien tietojen päätymisestä vain ja ainoastaan niiden kohteille. Koska datapalvelimen asiakasmoduulit tulevat olemaan omia ohjelmiaan, on niiden puskureita käytävä läpi mahdollisimman usein niiden täyttymisen estämiseksi. Tästä syystä puskuroijan toiminta ei saa olla hidastunut muun datapalvelimen yksikön toiminnasta, ja toisaalta puskureiden ollessa tyhjinä, niitä ei haluta olla tarkastamassa turhaan. Näistä syistä puskuroija kannattaa toteuttaa omana säikeenään niin, että se nukkuu puskureiden ollessa tyhjinä.

6.5. Käsittelijä

Kaikki yhteydet asiakaskomponentteihin tapahtuvat käsittelijän kautta. Se varmistaa oikeudet kaikkiin sen kautta suoritettaviin toimenpiteisiin tietoturvaajan avulla. Käsittelijä vastaa datapalvelimen yksikön omiin puskureihin tulleiden tietojen

käsittelystä. Sen tulee hoitaa myös J&T-rekisterin tietojen ylläpito. Rekisteriin kirjattaessa tulee tarkastaa, ettei tieto tule sinne kahteen kertaan: puskuri lisätään tilauksen ohjauspisteeksi vain tilauksen tekijän puuttuessa sieltä aiemmin, eikä uutta tilausta luoda kuin sen kokonaan puuttuessa. Käsittelijän tulee myös valita sellaiset kohteet tietojen välittämiselle, ettei verkkoon kohdistu liikaa rasitusta. Tähän hyödynnetään reitityskerroksen välittämiä tietoja.

Näin datapalvelin tilaa esimerkiksi tietyn lämpötilan vain kerran, vaikka sitä tilaisi kolme erillistä asiakasta erilaisissa muodoissa (esimerkiksi kelvin-, celsius- ja fahrenheit-asteina) – riippumatta itse julkaisussa käytetystä lämpötilan yksiköstä. Kun tätä tietoa sisältävä lähetys lopulta saapuisi tällaisessa tilanteessa käsittelijälle, ohjattaisiin se ensin liittäjälle kunkin asiakkaan tilaustietojen kanssa. Liittäjä voi mahdollisesti tarvita usean lähetyksen saadakseen tiedot tilattuun muotoon, joten jokaisesta lähetyksestä ei tule tietoja toimitettavaksi. Vasta tilatussa muodossa palautetut tiedot ohjataan niitä tilanneen asiakkaan puskuriin.

Käsittelijän tulee voida välittää asiakaskomponenteille virheilmoitus, mikäli se havaitsee niiden toimineen väärin (sen on todennäköisesti hyvä varoittaa tästä myös itse datapalvelimen yksikön operoijaa). Tällainen virheilmoitus tulisi lähettää myös silloin, kun tiedon käsittelemisen mahdollisuuksia ei ole – eli esimerkiksi jos tietoturvaaja ilmoittaa siltä puuttuvan tietyn luottamuksellisuusasteen vaatimuksiin kuuluvan salauskomponentin. Tietoturvaajan tehtäviin kuuluu kirjoittaa kaikista vastaavista tilanteista tieto lokiin, ja käsittelijän vastuulla on ilmoittaa tilanteesta.

Käsittelijä pitää yllä erillistä rekisteriä asiakasmoduulien liitoksista eri tilauksien muotoihin. Tämän rekisterin avulla se varmistaa tietojen ohjautuvan oikeille tahoille oikeissa muodoissa. Vasta viimeisen näistä aiemman esimerkin kolmesta tilaajasta lopettaessa tilauksen, poistuisi varsinainen tilaus. Käsittelijän pitää siis lähetyksiä saadessaan ensin selvittää, kenelle kaikille se on menossa. Muuten käsittelijä ei voi kertoa liittäjälle kuhunkin asiakasmoduuliin liittyvää tilausta – samaa tietoa voidaan tilata erilaisiin tilauksiin.

Kaiken asiakaskomponenttien kanssa tapahtuvan tietojen vaihdon tulisi tapahtua lähetys- ja vastaanottopuskureiden kautta, lukuun ottamatta varsinaista datapalvelimen yksikön piiriin liittymistä – puskurit voidaan luoda vasta tuon tapahtuman yhteydessä, joten siihen niitä ei voida käyttää. Näiden puskureiden koko voidaan asettaa riippuvaiseksi kunkin datapalvelimen yksikön asetustiedostossa olevista arvoista, ja niissä tulisi hyödyntää prioriteetin mukaista käsittelyä. Heikoimman prioriteetin omaava tieto ei ole yhtä tärkeä kuin muut, joten se voidaan poistaa puskurista sen ylivuotaessa. Tämä tieto voidaan esimerkiksi tallentaa välitysmuistiin talteen, jolloin puskuriin vapautuessa lisää tilaa sen syöttö sinne takaisin olisi helppoa.

Käsittelijän toimenkuvaan kuuluu siis tietojen vastaanottaminen muilta datapalvelimen yksiköiltä, sekä vastaanotettujen tietojen ohjaaminen niitä tarvitseville tahoille. Tällaisia tahoja voivat olla joko datapalvelimen omat asiakkaat (käsitelty yllä) tai muut datapalvelimen yksiköt (jolloin se lähetetään verkossa eteenpäin jäljellä

oleville tahoille). Myös nämä reitityskerroksen kautta lähtevät tiedot tulee kierrättää tietoturvaajan kautta ennen kuin niille tehdään mitään.

Toisten datapalvelimen yksiköiden oikeelliseen toimintaan tulee kuitenkin voida luottaa, eli toimenpiteiden oikeellisuus tarvitsee varmistaa vain kunkin datapalvelimen yksikön omalta osalta. Tällä tarkoitetaan vastaanotetun lähetyksen hyväksymistä oletusarvoisesti. Mahdollinen muille datapalvelimen yksiköille lähettäminen tarkistutettaisiin kuitenkin tietoturvaajalta.

Käsittelijä voi optimoida toimintaansa esimerkiksi pitämällä tiedossa joitakin tarkastussummia lähetysten kohdeosoitteesta ja lähetyksen puurakenteesta. Niiden perusteella se voisi ohjata lähetyksen suoraan samoihin puskureihin, jos siinä olevat lähettäjä-, kohde- ja tilaustiedot ovat edelleen samat. Myös tässä tapauksessa kysely tulee kuitenkin ajaa tietoturvaajan kautta. Näin voidaan varmistaa tiedon pysyminen vain luottamuksellisuusasteen mukaisella reitillä, vaikka jokin siihen vaikuttavista tekijöistä muuttuisikin. Esimerkkinä tällaisesta muutoksesta voisi toimia datapalvelimen ulkopuolisen avainten hallinnan tekemä päätös tiedon salaamiseen käytetyn avaimen kieltämisestä.

Lisäksi verkon toimintaa voidaan optimoida pitämällä datapalvelimessa omaa, reitityskerroksesta eriävää välimuistia välitetyistä tiedoista. Varsinaisen tiedoston hoitamisen vastuu on erillisellä tiedostojen hallinnan moduulilla. Välimuistissa olevan sisällön hoitamisen vastuu lasketaan kuitenkin datapalvelimen tehtäviin. Parhaiten sitä voidaan pitää yllä käsittelijän toiminnan yhteydessä – sen tulisi kirjata sinne aina viimeisimmät saapuneet tietorakenteet. Välimuistin täyttyessä tulisi pyrkiä etsimään sellainen tieto, joka on jo lähetetty eteenpäin ja jossa on sekä vanhin aikaleima että pienin mahdollinen prioriteetti. tulee välttää tietojen katoamisen estämiseksi. Jos lähettämättömiä tietoja joudutaan poistamaan, tulee siitä tiedottaa asiakasmoduulia.

Vanhimman ja heikoimman prioriteetin tietoa etsittäessä tapahtuvassa vertailussa ei kannata käyttää aina tarkinta mahdollista aikaleimaa, vaan verrata prioriteetista riippuvalla tarkkuudella – esimerkiksi pyöristämällä alaspäin minuutin tarkkuuteen prioriteetin ollessa alhainen ja käyttämällä täyttä tarkkuutta vain korkeimman prioriteetin tiedoissa. Näin tärkeämmät tiedot säilyvät välimuistissa kauemmin kuin vähemmän tärkeät, mutta välimuistissa pidetään silti tuoreimpia mahdollisia tietoja.

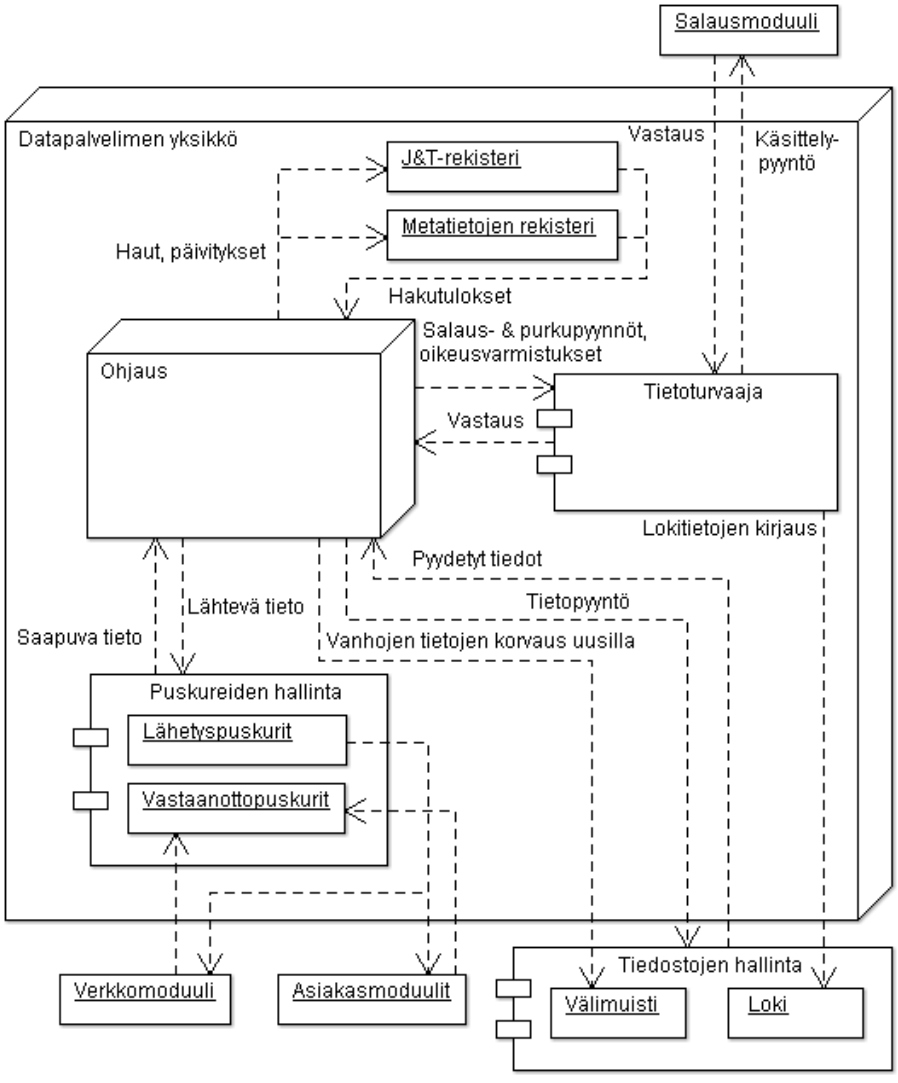
Välimuistin käsittelyyn liittyvät parametrit voidaan kertoa datapalvelimelle sen käynnistyksen yhteydessä asetustiedoston avulla. Näitä ovat paitsi sen koko (tyypillisesti satoja megatavuja), myös toimintatavan ohjaus (kuinka paljon tilaa millekin viestikategorialle vai ovatko kategoriat käytössä lainkaan, kirjoitetaanko aina vanhin tieto yli vai tarkastellaanko ensin prioriteettia, ja niin edelleen). Välimuisti parantaa ohjelmistoperheen nopeutta ja luotettavuutta oleellisesti, sillä solmujen väliset yhteydet voivat olla sekä epäluotettavia että hitaita.

Tämän hitauden vuoksi käsittelijän voi olla tarpeen jakaa iso lähetys pienempiin osiin, jotta mahdollisesti heikomman prioriteetin iso viesti ei pääsisi tukkimaan reittiä tärkeämmiltä lähetyksiltä liian pitkäksi aikaa. Käsittelijä toimii lähinnä tehtäviä jakavana toimijana. Koska muita osia on ohjattu omiin säikeisiinsä, ja käsittelijä ohjaa

niitä kaikkia, on käsittelijä luonnollinen valinta varsinaiseksi prosessiksi. Näin se voi helposti myös vastata ohjelman sulkemisesta ja tilanteen tallentamisesta ohjaamalla säikeet lopettamaan toimintansa sopivassa kohdassa niin, ettei minkään viestin käsittely lopu kesken vaan käsittelyyn otetut saadaan kaikki valmiiksi. Sen on myös luontevaa ohjata ne käynnistymään – jokainen muu datapalvelimen yksikön muodostavista moduuleista on siihen yhteydessä.

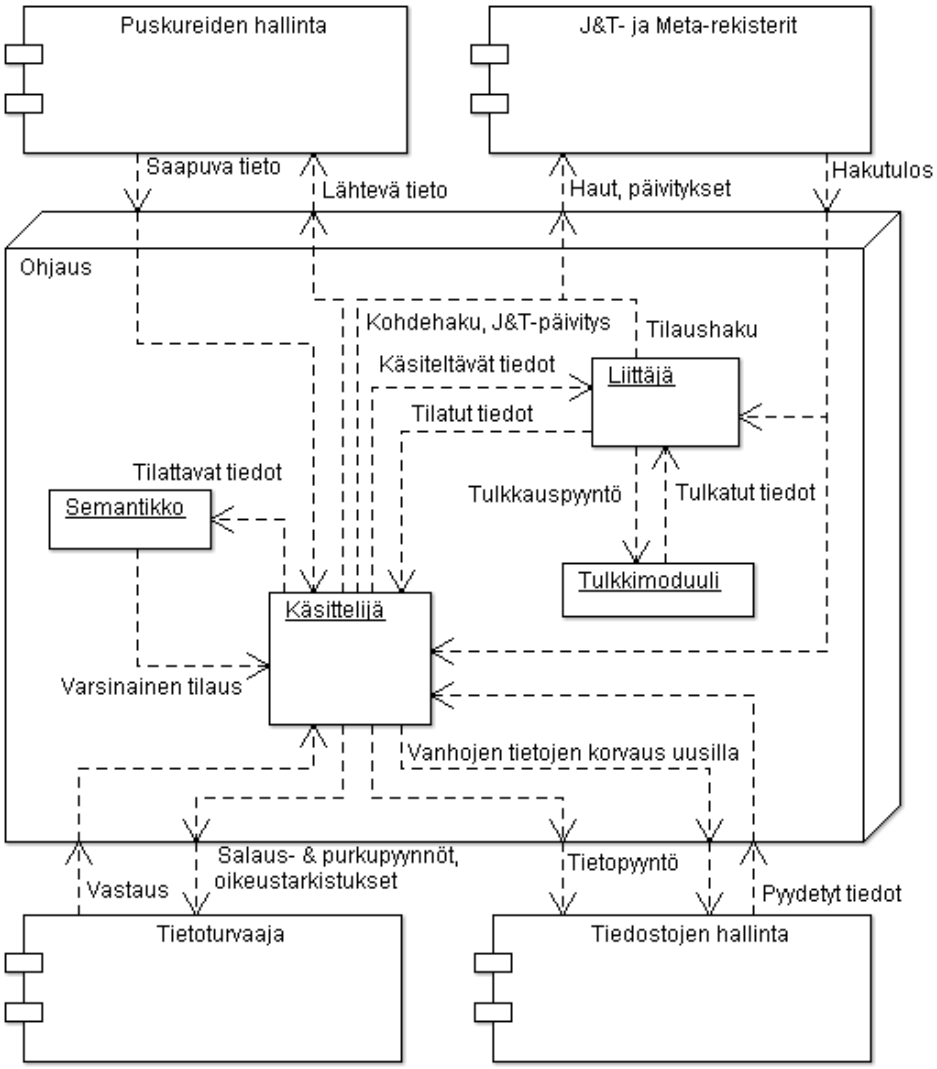
6.6. Osien yhteistoiminta

Varsinainen datapalvelimen yksikkö koostuu sen aiemmin esitellyistä osista. Sen kokonaisuuteen liittyvät tietovirrat on havainnollistettu graafisesti (kuva 6.2). Tietovirtojen kuvausta on yksinkertaistettu kuvaamalla siihen erillinen ohjausmoduuli. Tämä erottelu on tehty lähinnä kuvien selkeyttämiseksi, eikä varsinaista ohjausyksikköä ole tarpeen toteuttaa. Siihen liittyvät tietovirrat on eriytetty omaan kuvaansa (kuva 6.3). Kyseisen moduulin osalta ainoa datapalvelimen toteutuksen ulkopuolinen osa olisi tulkkimoduuli.



Kuva 6.2: Datapalvelimen yksikköön liittyvien tietovirtojen kuvaus

Käytännössä verkkomoduulina toimii datapalvelimen käyttöön otettu reitityskerros. Välimuistin toiminnasta vastaa niin ikään tiedostojen hallinnan moduuli. Salausmoduuli vastaa tietojen salaamisesta ja salauksien purkamisesta, ja sen avainten hallinnasta vastaa ulkopuolinen taho. Kahden viimeksi mainitun käytöstä vastuussa oleva tietoturvaaja hoitaa lokitietojen ylläpidon, jolloin myös salauksien purkamisesta voidaan kirjoittaa merkintä lokiin. Reitityskerros, salausmoduuli ja tiedostojen hallinnan moduuli ovat datapalvelimen varsinaisen toteutuksen ulkopuolella.



Kuva 6.3: Datapalvelimen ohjausmoduuliin liittyvät tietovirrat

Kuvat tuovat esille käsittelijän keskeisen aseman, sillä kaikki tiedot kiertävät aina sen kautta. Yksikön toiminta alkaa silloin, kun lähetys saadaan vastaanottopuskureista. Jos kyseessä oli tilaus- tai julkaisu-ilmoitus (tai jommankumman peruminen), kirjataan muuttuneet tiedot J&T-rekisteriin. Mikäli taas kyseessä on tietolähetys, kirjataan vastaanotetut tiedot välimuistiin. Jos välimuistille määritelty koko ei salli sinne kirjoitettavan lisää tietoa, kirjoitetaan vanhimman ja prioriteetiltaan heikoimman tiedon päälle uudempi. Ylikirjoitettavaa tietoa valittaessa kannattaa suosia jo lähetettyjä tietoja.

7. ASIAKASMODUULIN TAVAT KÄYTTÄÄ DATAPALVELINTA

Tässä luvussa esitetään datapalvelimen asiakasmoduulille tarjoamat käyttötavat. Ensimmäisessä kohdassa niistä kerrotaan yleisellä tasolla. Tämän jälkeen kokonaisuudesta eriytetään kaksi erillistä toimintaryhmää, joista kerrotaan kahdessa seuraavassa kohdassa.

7.1. Toiminta yleisellä tasolla

Asiakasmoduulit käyttävät datapalvelinta yleensä sen ohjelmointirajapinnan eli Application Programming Interface:n (jäljempänä API) läpi. Tämä API on luotu tarjoamaan asiakasmoduuleille kieli- ja kääntäjäriippumaton rajapinta datapalvelimen käyttöönottoa varten. Myös muut viestien välitykseen ja ohjelmistoperheen ytimeen kuuluvat osat tarjoavat käytetystä ohjelmointikielestä ja kääntäjästä riippumattoman rajapinnan API:en avulla.

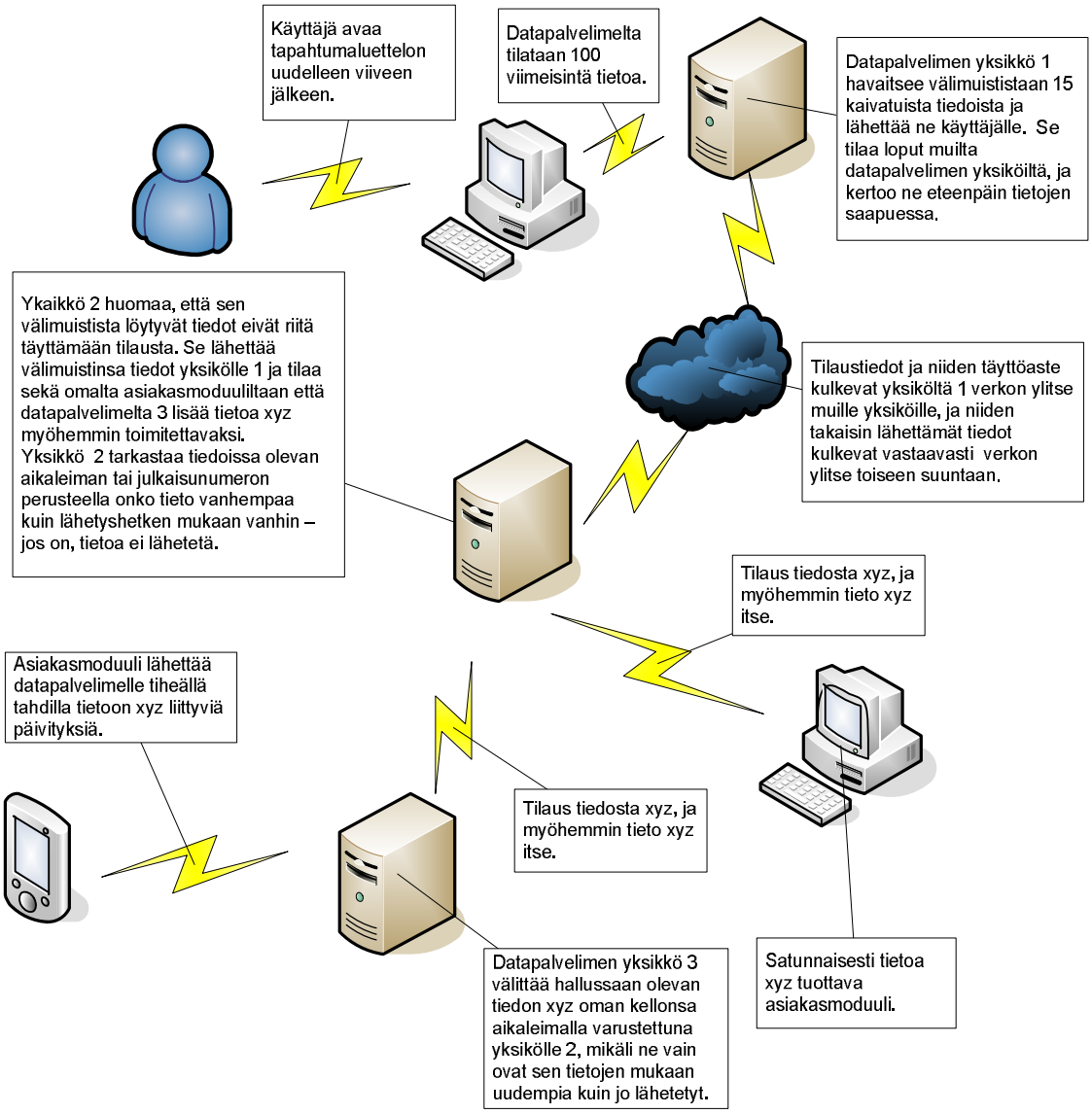
Asiakasmoduulit voivat käyttää datapalvelinta myös verkko- tai muiden yhteyksien avulla, esimerkiksi hyödyntämällä suoraan datapalvelimen alapuolista reitityskerrosta. Tällöin datapalvelin voi käsitellä saamansa viestit sellaiseen muotoon, että se voi kutsua omaa API:aan, tai API:n takana piilossa olevia funktioita – samalla estäen tarpeen käsitellä samankaltaisia tapauksia useissa kohdissa ohjelmaa. PSVSP:n ytimen käyttö ei näiden syiden vuoksi aseta projektin osille rajoituksia. Mahdollisuus käyttää sitä entistä monipuolisempien projektien yhteydessä onkin ollut eräs tavoitteista.

Datapalvelimen käyttötapoja voivat olla seuraavat: tietojen julkaiseminen tai tilaaminen sekä näiden toimintojen lopetus jonain myöhempänä ajankohtana. Näille toiminnolle on hyvä tarjota omat metodinsa rajapinnan toteutukseen. Tilaus ja julkaisu on eritelty kahdeksi erilliseksi kohdakseen (7.2 ja 7.3), joissa käsitellään myös peruminen. Yleisen tason toimintaesimerkki eräästä käyttötapauksesta on esitelty ohessa graafisesti (kuva 7.1, jonka tapahtumaketju alkaa käyttäjästä ja etenee siitä järjestyksessä eteenpäin) sekä liitteenä olevassa UML-sekvenssikaaviossa (liite 1).

Tietojen julkaiseminen datapalvelimen kautta tarkoittaa tietoa tuotettavan usein. Esimerkiksi julkaisu tuulen suunnasta ja nopeudesta ei tarkoita, että kyseessä olisi kertaluonteinen tieto, vaan että julkaisijan kautta voi hankkia tuoretta tietoa tästä aiheesta. Mikäli tiedot ovat rakenteeltaan yksittäisiä ja julkaisu peruutetaan heti ensimmäisen tietoja sisältävän lähetyksen jälkeen, ei datapalvelin voi tuoda tehokkuusetua suuren metatietomäärän vuoksi.

Esimerkiksi viestien lähetyksen yhteydessä näin voisi pikaisella ajattelulla luulla käyvän. Mutta vaikka uusia viestejä tulisi harvoin, on kaikissa niissä kuitenkin

tyypillisesti samanlainen rakenne: otsikko, lähettäjä, kohderyhmä tai yksittäinen kohde sekä varsinainen viesti. Kerrottaessa tämä rakenne ei julkaisua ole tarpeen perua, ja uudet viestit voidaan lähettää samalla, metatietopuussa jo olevalla kuvauksella.



Kuva 7.1: Yleisen tason toimintaesimerkki datapalvelinten verkon tapahtumista

Asiakasmoduuli voi ilmoittaa tarvitsevansa varmistuksen tiedon vastaanottajalta. Varmistettavan tiedon välittämisen yhteydessä käytetään hyväksi kuittauksia. Tieto voi edelleen edetä jotain reittiä pitkin, ja tällöin reitin varrella olevien datapalvelimen yksiköiden tulee välittää edelliselle lähettäjälle omat kuittauksensa. Näin varmistetaan viestin ja sen varmistustarpeen vastaanoton tapahtuneen oikein. Kun viesti sitten vastaanotetaan sen asiakasmoduulille luovuttavassa datapalvelimen yksikössä, tulee koko ketjun läpi lähettää vielä varsinainen kuittaus.

Mikäli viestin ensimmäisenä vastaanottaneen datapalvelimen yksikön sille asettama aikaraja ylittyy, ei sen kannata heti lähettää tietoja uudestaan. Sen sijaan tulisi lähettää varmistuskysely eli lähetys, jossa viitataan kuittauksen vaativaan lähetykseen, mutta jossa ei lähetetä tietoja uudestaan. Lähettämällä aluksi varmistuskysely voidaan

varsinaiset tiedot lähettää kohteeseen uudestaan sitä lähinnä olevasta solmusta. Se on todennäköisesti edelleen parhaimman kohteelle tiedon vievän reitin varrella, eikä silloin tarvitse välittää tietoja koko reitin matkalta.

Varmistuskyselyn vastaanottavan solmun tulee tarkastaa onko sillä kyseistä tietoa. Jos tieto on kyseisellä solmulla, lähettää se varmistuskyselyn eteenpäin sille solmulle, jonka reitityskerros ilmoittaa olevan seuraava askel kohteelle. Jos varmistuskyselyn vastaanottajalla ei ole varmistettavaa tietoa, tulee sen pyytää varsinaiset tiedot edeltävältä solmulta. Tieto tulee tämän jälkeen lähettää reitityskerroksen mukaan parasta reittiä kohti alkuperäistä kohdetta. Mikäli ketjun jossain kohdin havaitaan alkuperäisen vastaanottajan olevan poissa verkosta, tulee välittää epäonnistumisesta kertova viesti alkuperäiselle lähettäjälle, jonka perusteella lähettäjä ja ketjun varrella olevat tahot voivat poistaa tilauksen (tai lähetyksen) omista J&T-rekisterin tiedoistaan.

7.2. Tilaukset ja niiden peruminen

Tekemällä tilauksen asiakasmoduuli ilmaisee kiinnostuksensa vastaanottaa lähetyksiä tilaamastaan aiheesta. Kuva 7.2 esittää ylätason kuvauksen tietojen tilaukseen liittyvistä tapahtumista graafisesti. Sen tapahtumat alkavat vasemmasta yläkulmasta, jossa jokin taho ilmoittautuu asiakkaaksi. Tietorakenteiden tilaaminen ja tilausten lopettaminen tapahtuu kutsumalla datapalvelimen API:n vastaavaa metodia tai välittämällä vastaava kutsu jotain muuta kautta. Lopettamiseksi kerrotaan tilauksen tunniste, joka voidaan kertoa tilausmetodin kutsun palautteena. Tilauksen epäonnistumisesta voidaan kertoa esimerkiksi palauttamalla epäkelpo tunniste.

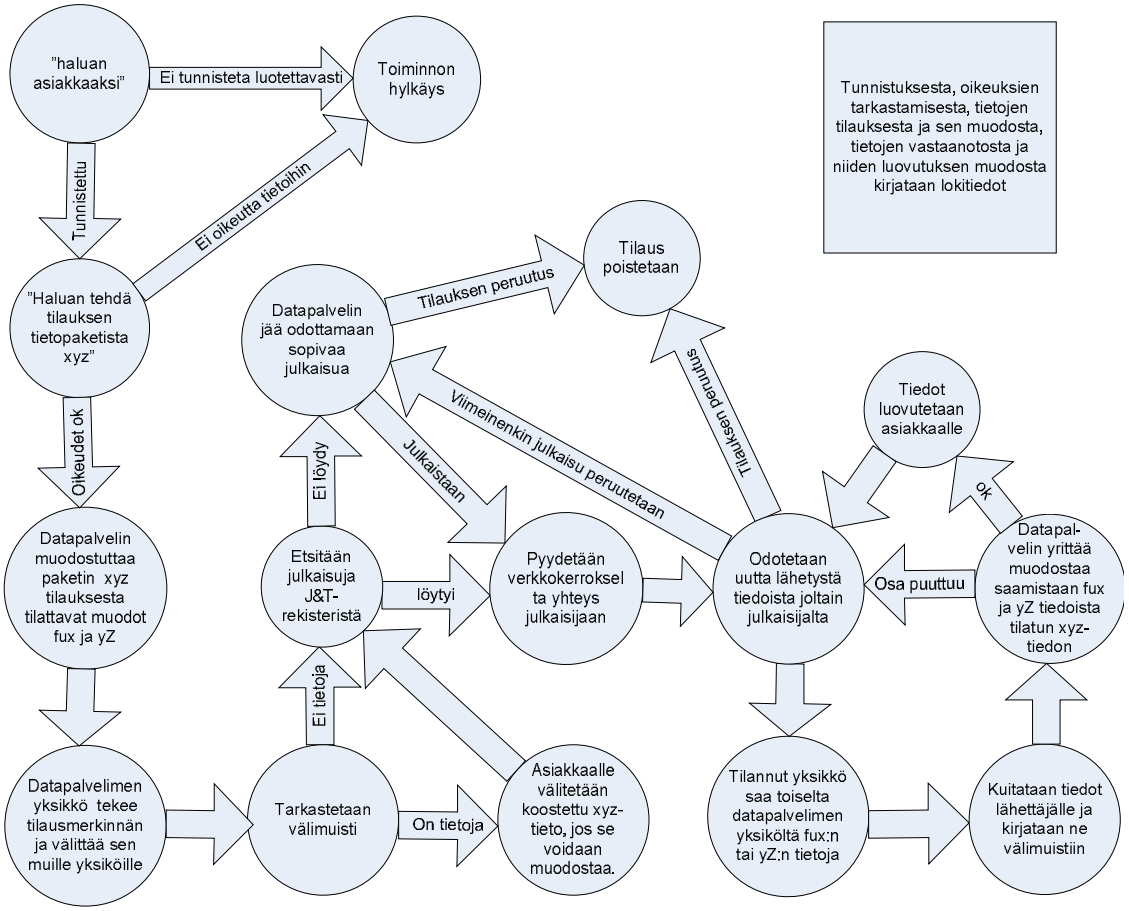
Tilauspyynnössä tulisi kertoa tietorakenteen lisäksi mahdolliset tietojen rajaukset ja yksiköt. Käytännössä nämä yhdessä kertovat mistä käsitteistä tilaaja on kiinnostunut ja missä muodossa asiakas haluaa toimitetut tiedot. Tarvittuihin tietoihin kuuluu myös salauksen ja sen avaimen tarjoaminen tilauksessa käytettäväksi. Niiden avulla asiakasmoduuli voi saada tilaamansa tiedot, vaikka sitä palveleva datapalvelimen yksikkö ei saisikaan käsitellä niitä salaamattomassa muodossa.

Lisäksi tulisi kertoa tilaustapa: halutaanko vain uusin tieto vai esimerkiksi 100 uusinta, ja kuinka usein lähetyksiä halutaan moduulille asti toimitettavaksi. Jos lähetysten aikaväliä rajoitetaan, tulee kertoa miten toimitaan jos tietoa tuotetaan useammin kuin sitä halutaan lähetettäväksi: tuleeko lähettää tietyin aikavälein vain uusin vai kerätäänkö vastaanotetut tiedot yhteen viestiin.

Tilauksen muotoilu on tilaajan vastuulla. Esimerkiksi ”100 uusinta lentokone ja nopeus -tietoparia” käsitteen tilaaminen ei aina ole riittävä kuvaus haluttujen tietojen saamiseksi. Ilman riittävää kuvausta tarjolla olevista vaihtoehdoista ei voida löytää oikeaa tietoa, ja asiakasmoduuli voi sen vuoksi saada vääränlaisia tietoja. Esimerkiksi nopeuden kuvauksen puute voi aiheuttaa sen, että asiakasmoduulille välitetään tiedot niiltä osin käsittelemättä.

Tämä nopeus voi siten olla esimerkiksi maanopeus, lentonopeus tai vaikka molemmat, riippuen tiedot julkaisseen moduulin tavasta kuvata niitä. Myöskään

yksiköstä ei voida tehdä oletuksia – vaikka saataisiin oikean tyyppinen nopeus, on maileina tunneissa ja kilometreinä tunneissa ilmoitetuilla lukemilla huomattavat erot, mikä johtaisi hyvin todennäköisesti epätoivottuihin tilanteisiin ja vääriin päätelmiin.



Kuva 7.2: Tilaukseen liittyvä datapalvelimen toiminta yhden asiakkaan suhteen

Todennäköisesti parempaan olisi voitu päästä kertomalla jonkinlainen rajaus tiedot kertovien lentokoneiden suhteen, ja että halutaan niiden maanopeus kilometreinä tunnissa. Näin tilaaja ei vastaanota ylimääräisten lentokoneiden tietoja, eivätkä nopeuden tiedot ole väärässä yksikössä tai muodossa. Ilman oikeanlaisia tietoja voitaisiin lentokoneelle virheellisesti ilmoittaa sen olevan lähestymässä liian nopeasti, kun luullaan nopeuden olevan maileina tunnissa sen ollessa kilometreinä tunnissa.

Kun datapalvelimen yksikkö saa tilauspyynnön, pitää sen pyrkiä tulkitsemaan pyyntö varsinaisiksi tilauksiksi. Tämän jälkeen yksikön tulee tarkastaa sen oman välimuistin tila sieltä mahdollisesti jo löytyvien lähetyksien varalta. Mikäli joitain kaivattuja tietoja tai niiden osia löytyy, niistä voidaan muodostaa halutussa muodossa oleva tieto – jos tässä onnistutaan, välitetään muodostettu rakenne tilauksen tehneelle kohteelle. Tämän jälkeen tulee kirjata mitkä kohdat kyseinen datapalvelimen yksikkö on jo kyennyt toimittamaan ja minkä lähetysten tiedoilla.

Tilaus voi sitten kulkea eteenpäin toisille datapalvelimen yksiköille, jotka koettavat vastaavalla tavalla täyttää vielä puuttuvat tiedot. Ne voivat myös toimittaa uudempia tietoja vanhempien sijalle. Eli myös muut datapalvelimen yksiköt tarkastavat, onko

niiden välimuisteissa uudempia tietoja jo lähetettyihin verrattuna, ja sen jälkeen katsovat onko J&T-rekisterissä tiedossa lähteitä pyydetyille tiedoille. Uudemmat tai puuttuvat tiedot lähetetään tilauksen tekijälle, jolloin ne päivittyvät myös matkan varrella olevien datapalvelimen yksiköiden J&T-rekisterin tietoihin.

Kun tilatuille tiedoille ei enää ole tarvetta, tulee tilaus perua tilaustunnisteen avulla. Näin verkossa ei turhaan pyritä siirtämään siihen liittyviä tietoja tai etsimään uutta julkaisijaa. Erityisen suuri merkitys perumiselle tulee tilanteessa, jossa tieto siirretään varmistettuna ja sitä eivät tilaa muut tahot. Tällöin tietoa ei tarvitse lähettää verkkoon, eivätkä useat solmut yritä varmistaa yhteyksiä kadonneeseen tilaajaan sen poistuessa verkosta. Mikäli tilaaja jää vielä verkkoon, aiheuttaa se turhaa resurssien tuhlausta myös itse tilaajan puolelta: sen pitää käsitellä ja hylätä turhat tiedot.

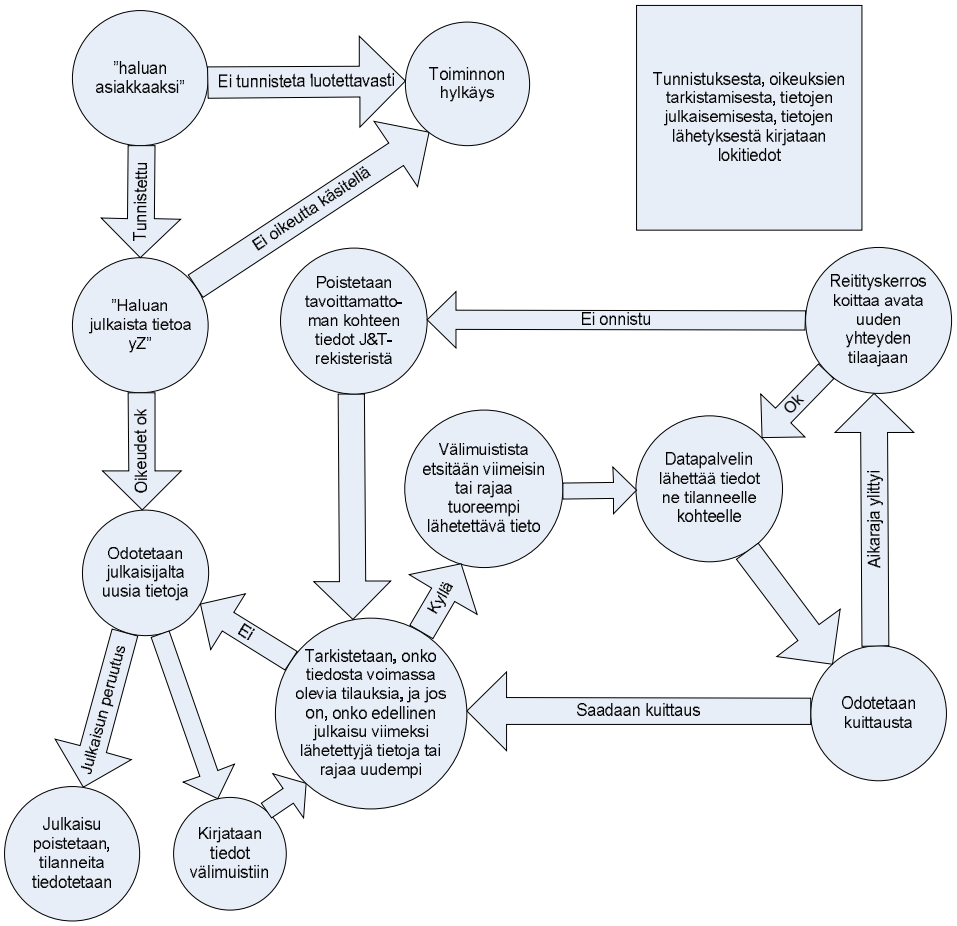
7.3. Julkaisut ja niiden lakkauttaminen

Julkaisemisella ilmoitetaan, että toimija tuottaa kyseisen kaltaista tietoa kunnes julkaisu peruutetaan. Toiminnoiltaan julkaiseminen eroaa tilaamisesta lähinnä sen suhteen, että julkaisut ovat valmiiksi oikeassa muodossa kun taas tilauksia voidaan joutua tulkitsemaan. Kuten vertaamalla kuvia 7.2 ja 7.3 voimme todeta, tulee julkaisusta siis yksinkertaisempi tapahtuma. Molemmissa kuvissa lähdetään tilanteesta, jossa jokin taho haluaa asiakkaaksi. Kuvaa on yksinkertaistettu huomioimalla vain yksi tilaaja, koska sama prosessi toistetaan jokaiselle tilaajalle.

Tietojen julkaiseminen tapahtuu kutsumalla datapalvelimen API:n julkaisun tarjoavaa metodia. Kutsussa kerrotaan datapalvelimelle kuvaus julkaistavaksi tarjotuista tiedoista. Mitä paremmin tiedot ovat kuvattuna metatietojen avulla, sitä paremmin datapalvelimen yksiköt kykenevät tarjoamaan julkaistuja tietoja niitä tarvitseville asiakaskomponenteille.

Julkaisutapahtuman päätteeksi datapalvelimen kannattaa palauttaa julkaisuun liittyvä tunniste. Tämän yksilöivän tunnisteen avulla on myöhemmin helppo hoitaa julkaisun lopettaminen. Varsinainen julkaisun lopettaminen tulee tarkoittamaan yleensä joko yhteyden katkeamista datapalvelimeen, laitteen sammuttamista tai esim. tietojen lähteen havainnoinnin lopettamista. Näistä vain yhteyden katkeaminen tarkoittaa virhetilannetta, josta datapalvelin toipuu lopettamalla kyseisen lähteen tietojen julkaisemisen.

Viimeisen lähetyksen päättymisen tai julkaistavan datan loppumisen jälkeen on osa normaalia toimintaa perua se hallitusti eli tiedottaa muutoksesta toiminnassa käytössä olevalle datapalvelimen yksikölle. Julkaisun peruvan ilmoituksen olemista osana laitteen tai moduulin hallittua sammuttamista suositellaan. Sekä hallitun että hallitsemattoman julkaisun lopettamisen yhteydessä voidaan lähettää viimeinen ”julkaisu”, jossa kerrotaan julkaisun päättymisestä ja tapahtuiko se hallitusti vai ei. Tämän tiedon avulla tilaajien on mahdollista erikoistaa reagointinsa tapauksien suhteen.



Kuva 7.3: Julkaisuun liittyvä datapalvelimen toiminta yhden asiakkaan suhteen

8. DATAPALVELIMEN ERI YKSIKÖIDEN VÄLINEN TOIMINTA

Tämän luvun alussa kuvataan datapalvelimen eri yksiköiden välinen toiminta tilanteessa, jossa niiden määrässä ja yhteyksissä ei tapahdu muutoksia. Tämän jälkeen esitellään toiminta uuden datapalvelimen yksikön liittyessä verkkoon ja vastaavasti yhteyden katketessa johonkin verkon solmuista. Erilliseen domain-tilaan liittyvä toiminta kuvataan seuraavaksi. Myös tilan tallennukseen liittyvän tilanteen hoito kuvataan omassa erillisessä alikohdassaan 8.5. Lopuksi kerrotaan kaksi lyhyttä esimerkkiä verkon toiminnasta eri tilanteissa.

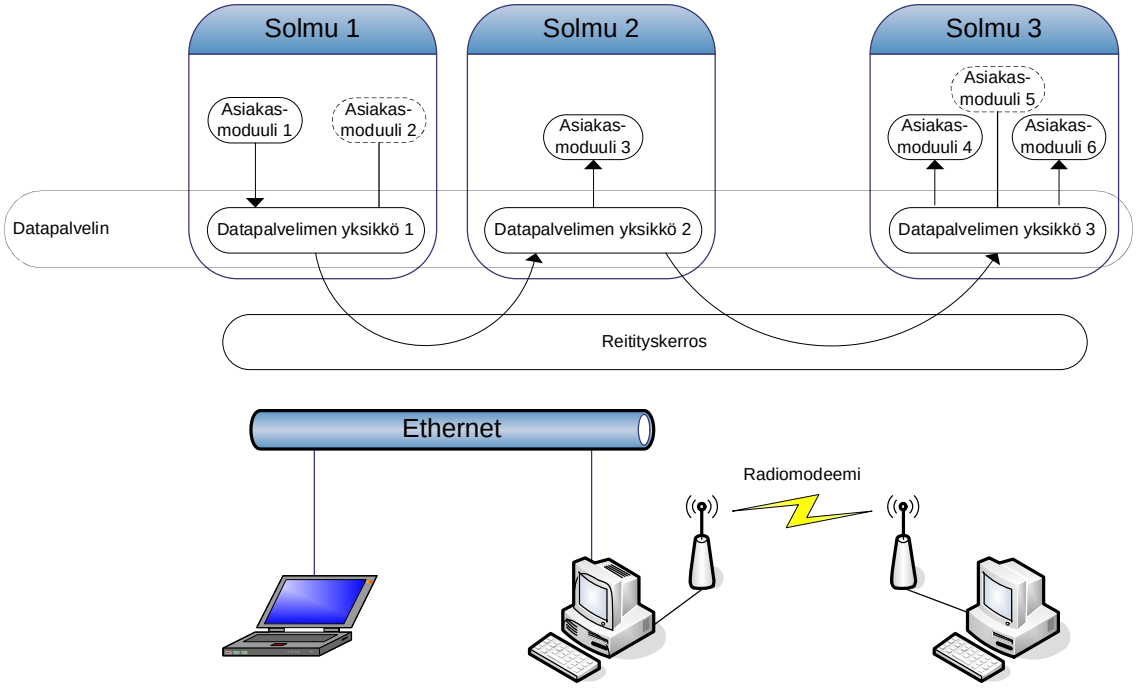
8.1. Ilman yhteyksien muutoksia solmuissa

Verkon stabiililla tilalla tarkoitetaan sitä, että solmut ja niiden yhteydet eivät koe muutoksia. Tällöin toimintaan aiheuttaa muutoksia vain asiakasmoduulien tilauksien ja julkaisujen muuttuminen. Pelkästään julkaisuja sisältävässä verkossa ei tulisi tapahtua varsinaista tietojen siirtoa, koska ilman siihen kohdistuvia tilauksia ei julkaistua tietoa ole syytä lähettää muille datapalvelimen yksiköille. Tällaisessa tilassa jonkin toisen datapalvelimen yksikön tekemä tilauspyyntö jostakin julkaistusta tiedosta käynnistäisi tietojen siirtämisen, täten moninkertaistaen verkossa tapahtuvan liikennöinnin.

Yksittäinen datapalvelimen yksikkö voi saada siihen kytkeytyneeltä asiakasmoduuliltaan toimintapyynnön joko tiedon tilauksesta tai julkistamisesta. Tieto tästä kannattaa ensin päivittää kyseisen datapalvelimen omiin tietoihin, jonka jälkeen nämä tiedot voidaan päivittää myös muille datapalvelimen yksiköille. Tilauksen välitystiedossa kulkee mukana sen varsinainen tilaaja, haluttu salaus ja avain sekä jo toimitetut lähetykset.

Välitystiedot kannattaa koostaa kirjoittamalla tiedolle useita toimitusosoitteita, mikäli samalle tiedolle on useita tilaajia. Kuva 8.1 esittää erään mahdollisen tilanteen, jossa asiakasmoduuli 1 tuottaa tietoa, jolle on kolme tilaajaa kahdessa erillisessä datapalvelimen yksiköissä. Siinä nuolelliset viivat kuvaavat tiedon siirtymistä eteenpäin, ja nuolettomat viivat kuvaavat tämän tiedon siirtämiseen tarpeettomia yhteyksiä.

Esimerkin tilanteessa solmulla 1 on tieto tilauksesta, jonka kohteina näkyy solmu 2 ja solmu 3. Mikäli jokin datapalvelimen yksiköistä havaitsee oman asiakasmoduulin tarvitseman tiedon tulleen julkaistuksi, pyytää se reitityskerrokselta suoraa yhteyttä julkaisevaan tahoon. Kuvan mukaisessa tilanteessa solmu 3 yrittäisi siis avata suoran yhteyden solmuun 1, ja epäonnistuessaan lähettää tilauspyynnön solmun 2 kautta – johtaen kuvassa esiintyvään tilanteeseen, jossa solmu 2 välittää tilauksen tiedot solmulta 1 solmulle 3.



Kuva 8.1: Tietojen siirtämisen ketjutuksen kuvaus

Tilaustietojen perusteella julkaiseva taho voi alkaa lähettää tilaajalle siltä puuttuvia tietoja. Datapalvelimen yksikkö solmussa 1 välittäisi siis esimerkin tilanteessa tiedot niille solmuille, joiden tilaukset sillä on tiedossa. Sillä ei ole tietoa monelleko asiakasmoduulille kukin datapalvelimen yksikkö välittää tietoa. Se ei voi yrittää ottaa suoraa yhteyttä solmussa 3 olevaan datapalvelimen yksikköön, koska se ei välttämättä tiedä kyseisen solmun avaimia. Tilaajan tulee siis avata yhteydet ja tarjota salausta sekä avainta käytettäväksi. Julkaisija voi hylätä tarjotun salauksen riittämättömänä, tai siihen käytetyn avaimen epäluotettavana. Tällöin tilaaja voi halutessaan koettaa jotain toista paria saadakseen yhteyden muodostettua. Yrityksiä ei kuitenkaan tule sallia loputonta määrää.

8.2. Datapalvelimen yksikön liittyminen verkkoon

Kun uusi datapalvelimen yksikkö yhdistää itsensä johonkin toiseen verkon datapalvelimen yksikön omaavista solmuista, kertoo reitityskerros uuden reitin avautumisesta yhteyden molempiin päihin. Tämän jälkeen datapalvelimet voivat vaihtaa oman J&T-rekisterinsä tiedot päivittääkseen ne, sekä propagoida muutokset muuhun verkkoon omille puolilleen. Käytännössä tämä päivitys kannattaa lähettää ensin koosteina eli tarkastussummina, joilla voidaan erottaa muuttuneet ja entuudestaan tunnetut puut toisistaan. Tällaisten tarkastussummien laskentaan voidaan käyttää tavanomaisia hajautusfunktioita, tai esimerkiksi Whirlpool-algoritmia [21, kohta 4.4.1].

Kuten luvussa 5.4 kerrottiin, eivät tavanomaiset hajautusfunktiot kuitenkaan ole yhtä luotettavia kuin kryptografisesti vahva algoritmi, vaikka ne ovatkin huomattavasti nopeampia (tavanomaisten hajautusfunktioiden muutaman operaatiokäskyn sijasta joudutaan Whirlpoolia varten suorittamaan 73 operaatiokäskyä jokaista

tarkastussumman lähtöaineiston bittiä kohti). Valintaa tukee myös datapalvelimelle asetettu resurssirajoitus verkon suhteen sekä suoritus- ja muistiresurssien rajoittamattomuus – laskemalla vahvemmalla algoritmilla saavutetaan todennäköistä etua rajoitetun verkkoresurssin suhteen käyttämällä enemmän muistia ja prosessoriaikaa.

Jos kokonaisuudesta lasketut tarkastussummat poikkeavat toisistaan, huomataan J&T-rekistereiden olevan eriävässä tilassa. Entuudestaan tunnetuista julkaisuista ja tilauksista laskettaisiin tällöin tarkastussummat kullekin puulle erikseen ja vastaavat tarkastussummat pyydettäisiin toiselta osapuolelta. Niiden saapuessa näitä verrattaisiin, ja mikäli ne olisivat yhtä suuret, tarkoittaisi se puiden olevan samanlaisia: varsinaista puuta ei tarvitsisi siirtää.

Vastaavasti mikäli tarkastussumma eriäisi, tai tietyn nimistä puuta ei entuudestaan tunnettaisi, kysyttäisiin vastapuolelta puun tarkempi kuvaus. Tämä olisi näistä kahdesta epätodennäköisempi tapahtuma, sillä julkaistavien tietojen rakenteen ei odoteta muuttuvan eri julkaisijoiden välillä. Samaa nimeä ei kannata sallia olevan useilla erilaisilla tiedoilla, sillä tämä voisi nimien monikäsitteisyyden vuoksi aiheuttaa väärien tietojen lähettämistä asiakasmoduuleille. Esto voidaan tehdä ainoastaan tietämällä etukäteen, onko verkossa julkaistuna muita tietoja samalla nimellä. Tämä tarkastus tulee tehdä jo asiakasmoduulia suunniteltaessa, sillä yritys hakea tietoa verkosta ennen sen julkaisua ei välttämättä aina tavoita kaikkia kohteita – eikä se siis voi aina estää erilaisten tietorakenteiden nimien törmäystä.

Lähetettävissä tilaustiedoissa tulisi pitää mukana myös tietoa siitä, kenelle datapalvelimen yksikölle tietoja vaihtava datapalvelimen yksikkö on sen tilannut (vain jos se on eri kuin se itse). Tällöin kaikki datapalvelimen yksiköt eivät tilaa samaa tietoa itselleen voidakseen toimittaa sitä jollekin toiselle yksikölle, koska ne voivat nähdä kenelle se on oikeasti menossa – ilman tietoa useampi datapalvelimen yksikkö lisää saman tilauksen omiin tietoihinsa voidakseen toimittaa sen eteenpäin, mistä aiheutuisi huomattavasti turhaa liikennettä verkkoon. Se myös paljastaa vaihtoehtoisia reittejä kohteisiin, mikäli jokin yhteys lakkaa toimimasta. Mikäli kohteen lisäksi pidetään tiedossa tähän mennessä kuljetun reitin kustannusta, saadaan vaihtoehtoiset reitit järjestykseen niiden hyödyllisyyden mukaan.

Tämän avulla voidaan tilaus liittää osaksi sitä ketjua, jossa lisäyksestä muodostuvat kustannukset ovat pienimmät. Jotta tämä toimisi, tilattua tietoa jo tilaavien ketjun jäsenten ei tule kasvattaa tätä kustannustekijää. Tämä toisaalta johtaa lievään kustannusten nousuun. Lisäkustannukset voidaan kuitenkin olettaa halvemmaksi kuin tietojen välittäminen epäoptimaalisia ketjuja pitkin. Ketjutuksen toteuttamisesta on kuvattu esimerkki kohdassa 8.7.

Päivitykset pitää kuitenkin saada kaikkien datapalvelimen yksiköiden tietoon. Tilauspäivitystä ei kannata lähettää samalle solmulle uudestaan. Tältä voidaan välttyä osittain: koska päivityksen vastaanottanut datapalvelimen yksikkö tietää miltä yksiköltä tuo päivitys tuli, voi se kysyä tuon yksikön naapurit reitityskerrokselta – vertaamalla tietoa lähettäjän naapureista omiin naapureihin voidaan vähentää turhia

uusintalähetyksiä. Naapurin naapureiden kyselemisestä ja vertailemisesta taas tulisi todennäköisesti enemmän lähetyskustannuksia kuin sallimalla toistuvat lähetykset tässä tilanteessa. Tämä mahdollisuus lähetyksen toistuvuudesta naapureiden naapurien tapauksessa on havainnollistettu ketjutuksen esimerkissä kohdassa 8.7.

8.3. Yhteyden katketessa toiseen yksikköön

Datapalvelimen ei itse kannata yrittää valvoa yhteyksien toimivuutta, vaan vastuu tästä voidaan delegoida OSI-mallissa alaspäin. Koska näiden kerrosten oletetaan voivan vastata siitä parhaiten, tehtävän voidaan myös katsoa kuuluvan niille. Kun reitityskerros kertoo yhteyden katkeamisesta toiseen datapalvelimen yksikköön, tiedetään, ettei siihen voida luoda uutta yhteyttä – reitityskerros pyrkii automaattisesti avaamaan menetetyt yhteydet (onnistuessaan sen tulisi ilmoittaa muutoksesta datapalvelimelle, mikäli reitti muuttuu). Täten datapalvelimen yksikön tulee tarkastaa tuon yhteyden toisessa päässä olleeseen liittyvät julkaisut ja tilaukset. Tämän jälkeen se poistaa tällaisten julkaisujen ja tilauksien välityksen omasta J&T-rekisteristään.

Lopuksi sen tulee kertoa muille tietämilleen datapalvelimen yksiköille juuri uudistuneet tarpeensa, mikäli joitain tarpeita ei enää esiinny – tarve ei katoa, jos yksikkö on tietoa kahteen suuntaan välittävässä roolissa, ja vain toisen suunnan yhteys katkeaisi. Näin tietoja ei välitetä turhaan kyseiselle datapalvelimen yksikölle asti. Mikäli reittiä pitkin on jo tulossa lähetystä nyt tarpeettomasta tiedosta, pysähtyy se tällöin ensimmäiseen sen turhaksi havaitsemaan solmuun. Usein lähetettävän tiedon tapauksessa tämä pysähtymiskohta lähenisi jatkuvasti kohtaa, jossa tieto muuttuu tarpeelliseksi.

On huomioitavaa, että ainoastaan pudonneiden tapauksien tulisi poistua, eli jos esimerkiksi kyseiseen datapalvelimeen olisi sidoksissa sellainen asiakasmoduuli, joka kaipaisi samoja tietoja kuin jokin verkosta pudonneen datapalvelimen yksikkö, ei sen tilauksen tulisi poistua kyseisen yksikön tilausluettelosta – ainoastaan kadonneen kohteen tiedot poistettaisiin eteenpäin toimitettavien joukosta. Tällaisten yhteyksien katkeamisen oletetaan tapahtuvan pääasiassa väliaikaisesti. Tämän vuoksi datapalvelimen ei kannata pyrkiä aktiivisesti poistamaan tuntemiaan metatietoja heti yhteyden katkettua, vaan ne kannattaa pitää tallessa mahdollisimman pitkään. Näin verkon tehokkuutta voidaan parantaa välttämällä tarvetta siirtää näiden puiden kuvauksia uudelleen yhteyksien palautuessa.

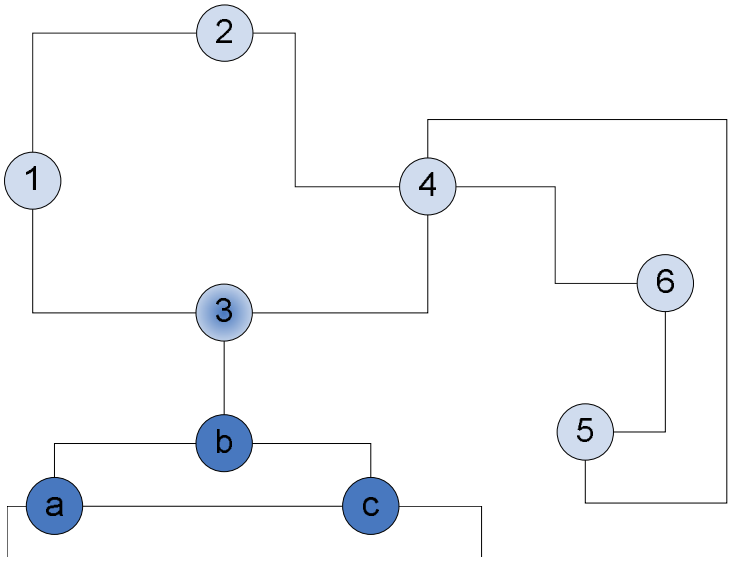
Mikäli tällainen yhteyden katkeaminen johtaa tilanteeseen, jossa datapalvelimen yksikkö on eriytynyt muusta verkosta, ei se aiheuta tarvetta poiketa tästä yleisestä tavasta toimia. Tällaisessa tilanteessa yksikkö toimii yksinään kuin datapalvelimen kokonaisuus: jos sillä ei ole toimittajia tilauksien tiedoille, ei se tee niille mitään. Vastaavasti jos sen kautta pyritään lähettämään tietoja, ne jäävät välimuistiin talteen. Kun yhteydet myöhemmässä vaiheessa mahdollisesti palaavat, tulee kyseiselle yksikölle tietoon myös muiden yksiköiden tilauksia, jolloin se voi lähettää välimuistissaan olevat tiedot ehdot täyttäville tahoille.

Tilanne voi muodostua ongelmaksi ainoastaan tilanteen pitkittyessä. Tällöin välimuisti voi täyttyä, ja osa tiedoista menetetään. Tämän vaikutusta voidaan pienentää jättämällä talteen vain tuoreimmat ja prioriteetiltaan tärkeimmät tiedot. Koska välimuistin koko on suuri, ei kaikkia siellä olevia tietoja voida toimittaa kerralla eteenpäin. Lisäksi osa tiedoista voi olla vanhentuneita. Tämän vuoksi välimuisti tulee purkaa prioriteetin mukaisessa järjestyksessä, ja lähettää tiedoista uusimmat ensin.

Näin todennäköisesti tärkeimmät tiedot saadaan toimitettua mahdollisimman nopeasti – esimerkiksi nykyisen tilanteen selittävä viesti on huomattavasti tärkeämpi kuin tuntia aiemman tilanteen selvitys, vaikka molemmilla viesteillä on sama prioriteetti. Kaikkein vanhimpia tietoja ei välttämättä ole koskaan tarpeen lähettää eteenpäin. Tällöin tietoa ei kuitenkaan kannata poistaa välimuistista, sillä joku muu asiakas voi olla kiinnostunut historiatiedoista. Tarve lähettää selviää tietoja tilanneiden tahojen tekemistä tilauksista, koska niissä kuljetetaan mukana tietoa yhä toimitusta vailla olevista tiedon lähetyksistä.

8.4. Domain-tilan hyödyntäminen

Domain-tilalla tarkoitetaan datapalvelimen yksikön yhteydessä sellaista tilaa, jossa jokin datapalvelimen yksikkö koostaa muille yksiköille sen kautta saavutettavissa olevat tiedot ja tarpeet. Kuva 8.2 esittää tilannetta, jossa domain-tilaa voidaan hyödyntää tehokkaasti. Kuvan solmu 3 voi koostaa numeroitujen solmujen tilaukset ja julkaisut, ja välittää ne kokonaisuutena solmulle b. Edelleen solmut a ja c voivat olla yhteydessä muihin domain-tilassa oleviin solmuihin, jolloin solmut a, b, c ja 3 yhdistävät kolme erillistä verkkoa toisiinsa.



Kuva 8.2: Verkkojen yhdistäminen toisiinsa Domain-tilan avulla

Kuvan numeroidut solmut voivat olla yhteyksissä toisiinsa esimerkiksi radioverkon avulla, kun kirjaimelliset taas ovat kiinteämpiä solmuja normaalissa lähiverkossa. Jos oletamme myös solmun 3 olevan lähiverkolla yhteydessä solmuun b, niin sillä on jo

valmiiksi luontainen rooli verkkoja yhdistävänä solmuna. Näin voidaan välttää pitkiä reittejä epäluotettavassa verkossa. Lisäksi tällaisten verkkoja yhdistävien solmujen avulla voidaan rajoittaa tilaus- ja julkaisutietojen aiheuttamaa liikennettä. Ne tuovat samalla hierarkkista rakennetta verkkoon.

Erityisen hyödyllinen niiden mukanaan tuoma rakenne on silloin, kun heikompia turvavaatimuksia sisältävän tason liikenne liikkuu luotettavampien yhteyksien päällä, tai toinen verkko on tarkoitettu luottamuksellisemman tiedon käsittelyyn. Jos ajattelemme solmujen a, b, c ja 3 välisen lähiverkon vaativan solmuiltaan korkean turvallisuusluokan käsittelyoikeudet, ei muiden numeroitujen solmujen tule kyetä viestimään kyseisessä verkossa ilman samoja oikeuksia [25, luku 8]. Koska solmujen tulee tietoturvaoppien mukaisesti toimia pienimmillä mahdollisella määrällä oikeuksia, voisi tämä johtaa sellaisten erillisten saarekkeiden muodostumiseen, joista tieto ei pääse etenemään.

Mikäli solmu 3 saa toimia näiden erillisten verkkojen välissä yhdistävänä tekijänä, voidaan epäluotettavamman verkon tiedot kuljettaa rajoittavamman luokituksen omaavassa verkossa. Tällöin ne voivat kyseisen rajoitetun verkon kautta olla yhteydessä muihin oman luokkansa verkkoihin. Tämä voidaan toteuttaa esimerkiksi pakkaamalla epäluotettavamman verkon tiedot erilliseksi domain-julkaisuksi, joita muut saman tason verkon domain-tilassa olevat yksiköt tilaavat.

Näin verkkoon muodostuvat saarekkeet eivät muodostu ongelmaksi eikä verkossa liiku niin paljoa tietoa solmun 3 tehdessä suodatusta välittämistään tiedoista. Luonnollisesti solmun 3 tulee kuitenkin estää rajoitetun luokan tiedon kulkeutuminen heikomman turvatason verkkoon, tai vaihtoehtoisesti solmulle tulee myöntää vain oikeus yhdistyä kyseiseen verkkoon. Näin muut kuin verkolle kuuluvat tiedot eivät tulisi solmulle 3 asti. Vastaavasti solmun 3 tulee asettaa domain-tilaan liittyvän liikenteen prioriteetti kaikkia rajoitetumman luokan tietoja alhaisemmaksi – näin vähennetään kykyä suorittaa kohdassa 4.3.3 kuvattu palvelunestohyökkäys verkkoa vastaan, koska hyökkäykseen liittyvät lähetykset olisivat aina ensimmäisenä tippumassa pois lähetettävien joukosta.

Domain-tila tulee kuitenkin voida ottaa pois päältä, koska on edelleen riski ylimääräisen liikenteen aiheuttamista toissijaisista vaikutuksista – esimerkiksi pyrittäessä pitämään liikennöinti mahdollisimman alhaisena voidaan domain-tilaan liittyvä liikenne haluta sulkea kokonaan pois. Mahdollisena syynä esimerkin tarpeeseen voi olla esimerkiksi tarve vaikeuttaa verkon olemassaolon havaitsemista. Vastaavasti liikennöintiä voidaan haluta rajoittaa myös tilanteissa, joissa lähetyksien suhteen on asetettu muita rajoitteita – esimerkiksi lähettimen sisältäessä vain rajallisen määrän virtaa akussaan, tulee tuo virta varata vain tärkeiden viestien lähettämiseen. Alemman luokan tietojen välitystarpeita ei tällöin voida huomioida, sillä ylemmän luokan verkon toimintakyky ja vaatimukset menevät niiden edelle.

Domain-tilalla vähennetään myös tietojen muuttamisen tarvetta. Jos solmun 3 takana olevan radioverkon sisällä tapahtuu yhteyksien katkeamisia usein, aiheuttaisi tämä mahdollisesti hyvinkin usein tapahtuvaa tarvetta päivittää J&T-tietoja datapalvelinten yksiköiden välillä. Koska voimme olettaa ainakin osan solmuista olevan

yhteydessä solmuun 3, voi se domain-tilassa toimiessaan jatkaa tietojen julkaisemista ja tilaamista sen takaisesta verkosta. Domain-tilassa toimiessaan datapalvelimen yksikön voidaan sallia pitää tilaus tai julkaisu voimassa myös ilman suoranaista tilausta tai julkaisua, koska se olettaa niiden voimassaolon palautuvan hetkittäisen yhteyden katkeamisen jälkeen.

Näin se peittää ja rajoittaa muutosten tekemisen tarvetta paremmin kuin tavallisessa tilassa toimiva datapalvelimen yksikkö. Toimintatapa auttaa erityisesti silloin, kun domain-tilassa olevan datapalvelimen yksikön takana on useita julkaisuiltaan ja tarpeiltaan samantapaisia solmuja, eivätkä J&T-rekisterin tiedot vaihdu usein. Tälle toiminnalle on kuitenkin asetettava aikaraja, jotta verkkoon ei aiheuteta turhaa liikennettä siirtämällä tietoja turhaan domain-tilassa olevalle solmulle. Samasta syystä myös J&T-rekisterin tietojen muuttumisen muusta kuin yhteyden katkeamisesta johtuvista syistä tulee kuitenkin mennä tämän olettamuksen edelle.

8.5. Tilannevedoksen luonti ja tilan palautus

Datapalvelimelle asetettuihin vaatimuksiin kuuluu myös mahdollisuus tallentaa palvelimen tila, sekä myöhemmin palauttaa se tällaisesta tallenteesta. Koska tiedot voivat olla turvallisuuskriittisiä, tulee kaikki kirjoitettu tieto olla salatussa muodossa. Kun tietyn datapalvelimen yksikön tilanne pyydetään tallentamaan, sen toiminta kannattaa pysäyttää verkon viestinnän kannalta – näin vältetään ongelmia tietojen synkronoinnista. Tästä kannattaa ilmoittaa asiakasmoduuleille viimeisenä toimintona ennen liikenteen pysäyttämistä. Näin ne eivät ajaudu virhetilaan. Muut datapalvelimen yksiköt saavat tiedon tilanteesta yhteyksien katkeamisen kautta.

Varsinainen tietojen tallennus voidaan tehdä datapalvelimen ulkopuolisen tiedostojen hallintaa hoitavan moduulin avulla. Se vastaa kokonaisuudessaan tiedoston salaamisesta sekä sen eheydestä. Sen kautta kirjattaisiin paitsi tunnetut datapalvelimen yksiköt, myös näihin liittyvät J&T-rekisterin tiedot. Niin ikään tiedot datapalvelimen yksikköön suoraan yhteydessä olevista asiakasmoduuleista sekä näihin liittyvät J&T-rekisterin tiedot tulee tallentaa. Myös välimuistissa olevat tiedot kannattaa kirjoittaa talteen, sillä joidenkin tietojen eteenpäin lähettäminen voi olla tarpeellista tilanteen palautuksen yhteydessä.

Datapalvelimen yksikön voidaan odottaa liittyvän takaisin verkkoon kohtuullisen lyhyessä ajassa, jolloin suurin osa sen välimuistin tiedoista olisi todennäköisesti yhä ajan tasalla. Välimuistien tietojen mukaan ottaminen auttaa myös parantamaan verkon toimintaa niissä tilanteissa, joissa kriittisiä tietoja on jäänyt talteen vain juuri toimintansa tallentavan datapalvelimen välimuistiin. Tällaisesta toiminnasta ei ole haittaakaan, koska uudelleen lähetettävät lähetykset voidaan estää tunnistamalla ne tilaustiedoissa olevista aiemmista lähetyksistä.

Tilan palautus voidaan aloittaa joko datapalvelimen yksikköä käynnistettäessä tai vasta myöhemmässä vaiheessa käynnistyksen jälkeen – koska tänä aikana se on jo voinut saada uusia tietoja, ei vanhoja voida tässä tapauksessa ottaa suoraan käyttöön.

Tällöin palautustiedoston tiedot lisätään J&T-rekisteriin ja välimuistiin jätetään tuoreimmat sinne mahtuvat tiedot. Palauttamisen aikana tulee pyytää reitityskerrosta muodostamaan yhteys niihin kohteisiin, joihin oltiin sen kautta yhteydessä tilaa tallennettaessa. Mikäli yhteyttä johonkin näistä ei saataisi muodostettua uudestaan, tulee toimia kuin yhteys siihen olisi juuri katkennut.

Omien J&T-rekisterin tietojen kertominen on osa uuden yhteyden onnistunutta avaamista. Kun tilaa palautetaan, ei sen tila kuitenkaan ole ajan tasalla. Niinpä uutta tilaa ei kannata kertoa muille datapalvelimille ennen kuin oma tila on saatu kokonaisuudessaan päivitettyä. Tämän vuoksi datapalvelimen yksikön tulee voida kertoa toiselle yksikölle, että se on palautus-tilassa. Näin ei päädytä kertomaan mahdollisesti vääriä tietoja muille datapalvelimen yksiköille. Kun palautus on saatu valmiiksi, tulee tietojen vaihto toteuttaa loppuun asti – eli kertoa naapuriyksiköille oma J&T-rekisteri. Tämä voidaan tehdä käymällä koko vaihtoprosessi uudestaan läpi. Tästä saavutetaan etua, mikäli jonkin muun tila on ehtinyt tässä vaiheessa päivittyä.

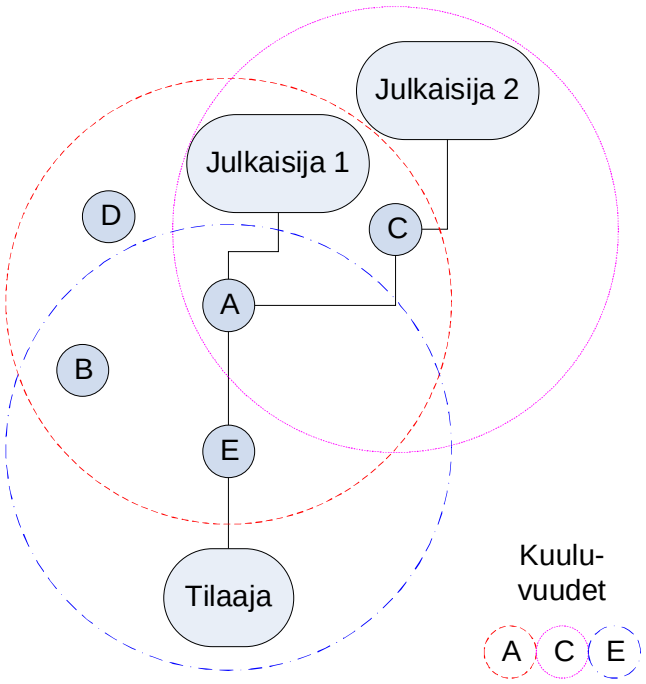
8.6. Esimerkki usean tietolähetyksen tilaamisesta radioverkossa

Tässä kohdassa on ensin yleinen kuvaus tilanteesta. Tämän jälkeen kuvataan radioverkon erityispiirteisiin liittyvien tarpeiden huomiointi. Lopuksi kuvataan ne seikat, joiden huomiointi on tarpeen useiden samaan tietoon liittyvien tietolähetyksien tilauksien yhteydessä.

8.6.1. Tilannekuvaus

Oletetaan esimerkin vuoksi viisi radiotiellä viestivää datapalvelimen yksikköä nimitetyn A:sta E:hen. Oletetaan lisäksi, että A:han ja C:hen on kiinnittynyt tahoja, jotka julkaisevat sellaista tietoa, josta jokin E:hen kiinnittynyt taho haluaa sata tuoreinta tietoa itselleen. Näille tiedoille reitityskerros kertoo lyhimmäksi reitiksi kuljetuksen A:sta E:hen, ja siitä eteenpäin varsinaiselle kohteelle. Tällöin A tilaa C:ltä näitä tietoja. Kuva 8.3 esittää tilanteen graafisesti kuvaamalla viivoilla käytettäviä yhteyksiä ja katkoviivoitetuilla ympyröillä radioiden kuuluvuuksia.

Verkossa on myös muita mahdollisia yhteyksiä (esimerkiksi solmu A voi viestiä kaikille esitetyille solmuille, ja solmu E voi olla yhteydessä solmuun B). Ne jätetään huomiotta, jotta esimerkkitilanteesta saadaan selkeämpi. Näillä huomioimattomilla yhteyksillä ei ole esimerkin tapaukseen liittyvää liikennettä.



Kuva 8.3: Kuvaus radioverkkoesimerkin tilanteesta

8.6.2. Radioverkon erikoispiirteiden huomiointi

Kun tietoja lähetetään esimerkiksi radioteitse, sen voi kuulla usea datapalvelimen yksikkö. Datapalvelimen yksiköiden toiminnalle tästä voisi tulla ongelmia, koska se voi johtaa saman lähetyksen moninkertaiseen välittämiseen. Tähän voidaan kuitenkin varautua reitityskerrosta suunniteltaessa esimerkiksi kertomalla aina viestissä paitsi sen lähettäjä ja kohde, myös kenelle se on tällä kertaa tarkoitettu (eli mikä on seuraava solmu sille osoitetun reitin varrella). Tämän vuoksi muut datapaketin kuulleet, ylimääräiset tahot jättäisivät sen huomiotta, eikä se saapuisi datapalvelimen kerroksen tasolle asti. Tilanteeseen voidaan varautua verkkokerroksessa tehokkaammin kuin datapalvelimen tasolla.

Esimerkiksi solmujen B ja D reitityskerroksen osat voivat kuulla A:n lähetyksen E:lle, mutta ne eivät käsittelisi tietoa sen osoituksen kertoessa kohteen olevan jokin muu solmu. Ne eivät siis koskaan saavu solmuissa oleville datapalvelimen yksiköille. Tiputtamisen syynä on tarve mahdollisimman vähäiseen siirtotien käyttöön. Tämä on erityisen tärkeää yhteyden toimiessa radioverkon yli, koska siirtokapasiteetti voi olla erittäin rajoittunutta. Jos kaikki lähetyksen kuulleet reitityskerroksen osat lähettäisivät sen eteenpäin (ja toistuvat lähetykset tiputettaisiin heti sellainen havaittaessa), olisi verkon siirtotietä kulutettu liikaa.

Tämä vähentää verkon toimintavarmuutta tilanteissa, joissa solmujen yhteys toisiinsa on epävarma. Reitityskerroksen on siis huolehdittava lähetysten toimittamisen varmistamisesta. Tämä tehtävä voidaan hoitaa esimerkiksi kuittausviestien avulla. Niiden käytöllä aiheutetaan huomattavasti vähemmän rasitusta verkon siirtokapasiteettiin, kuin sallimalla usean tahon lähettää toisille tarkoitettu viesti

eteenpäin. Erityisen ilmeistä tämä on tilanteissa, joissa toisen solmun liikennöinti estää muiden solmujen liikennöinnin – mikä on yleistä radioverkoissa, joissa lähetyksen aktivointi jollakin taajuudella estää muita kuuluvuusalueen käyttäjiä hyödyntämästä samaa taajuutta. Ratkaisuna edelliseen olisi käyttää eri yhteyksille eri radiotaajuuksia, mutta tähän ei aina ole mahdollisuutta.

Vaikka haasteen ratkaiseminen ei sisälly datapalvelimen tehtäviin, sen on kuitenkin huomioitava tilanne omassa toiminnassaan. Tämä näkyy erityisesti myös datapalvelimeen ulottuvassa tavoitteessa karsia verkon liikenne mahdollisimman vähiin. Tavoite on myös datapalvelimeen valittujen teknologioiden (eli vertaisverkko- sekä julkaisija–tilaaja-mallin) taustalla.

8.6.3. Usean tietolähetyksen tilauksen huomiointi

A voi rajoittaa E:lle lähetettyjä tietoja sen mukaan, mitä se tietää lähetetyn eteenpäin. Näiden rajoitusten avulla voidaan estää lähettämästä sataa viimeisintä tiedossa olevaa lähetystä vanhempia tietoja. Jos tiedot ovat yleisluonteisia eli eivät ole tuottajasta riippuvaisia, ei E tietäisi osan tiedoista tulevan C:ltä. E tietäisi vastaanottamiensa lähetyksien perusteella ainoastaan kuinka suuri osa sen tilaamista tiedoista on jo löytynyt.

Mikäli A:n lähettämät viestit eivät tavoittaisi E:tä, ilmoittaisi reitityskerros yhteyden katkenneen. Koska reitityskerros pyrkii tällaisessa tilanteessa automaattisesti muodostamaan uuden reitin kohteeseen, tarkoittaa ilmoitus yhteyden katkeamisesta tässä epäonnistumista. Tämän vuoksi A:n tulee poistaa ne J&T-rekisterin tiedot, jotka liittyvät yhteyden päässä olleisiin kohteisiin. Kun yhteys mahdollisesti myöhemmin palautuu, todennäköisesti myös tietojen tilaus tulee taas voimaan. Mikäli välimuistissa olevat tiedot ovat tuolloin vielä ajankohtaisia, lähetetään ne tämän jälkeen tilaajalle. Ajankohtaisuuden päättää rajaus, joka suoritetaan lähetysten tunnisteiden perusteella.

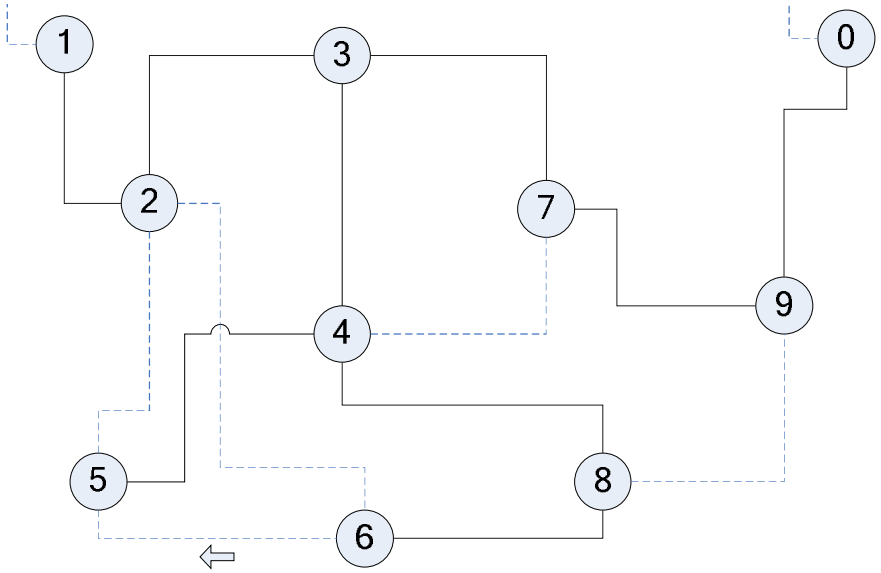
Tämä raja muodostetaan tilauksen perusteella. Esimerkissä tiedoista on tilattu sata uusinta, ei sataa ensimmäistä löydettyä tietoa. Tilausta ei siis lopeteta ensimmäisen sadan tiedon toimittamisen jälkeen, vaan tilaajalle lähetetään sata tilaushetkellä uusimmaksi todettua sekä sen jälkeen ilmestyvät tiedon lähetykset – käytännössä siis tällaisella määrällisellä tilauksella ilmaistiin halu saada myös vanhempia tietoja kuin uusin ja sen jälkeen tulevat. Sadanneksi vanhimpana olevan tietojen lähetyksen katsotaan toimivan rajana, jota uudemmat viestit toimitetaan tilaajalle – mikäli niitä ei vielä ole lähetetty.

Jotta vältetään samojen tietojen lähettäminen useaan kertaan, ei kuitenkaan voida suoraan lähettää varsinaisia tietoja. Sen sijaan tulee ensin lähettää tilaajalle ilmoitus, jossa kerrotaan minkälaisia tietoja kyseiseltä datapalvelimen yksiköltä voi saada. Alustavana karsintana toimii tilauksessa jo lähetettyjen tietojen lista, jonka avulla vältetään kertomasta jo toimitettujen tietojen saatavuudesta. Tällöin tilaajana toimiva datapalvelimen yksikkö voi valita sopivat lähteet tiedoille, ja pyytää näitä lähteitä välittämään tiedot.

Lähteiden valinnan apuna voidaan hyödyntää reitityskerrosta sekä sen tietoja reittien kustannuksista. Kun tällainen välitys hyväksytään ja tiedot lähetetään, reitin varrella olevat muut yksiköt voivat päivittää J&T-rekisterin tietonsa kyseisen tilauksen osalta. Vastaavasti hylättyjen reittivaihtoehtojen datapalvelimen yksiköille kannattaa myös lähettää viesti, jossa kerrotaan tiedon saannista muuta kautta. Tällöin ne voivat päivittää tilaustietonsa kyseiseltä osalta.

8.7. Tilausten ketjuttamisen esimerkki

Kuva 8.4 esittää graafisesti datapalvelimen kymmenen eri yksikön muodostaman verkon, jossa kukin yksikkö muodostaa yhden solmun. Solmuista 1 ja 0 voidaan olla yhteydessä muihin verkkoihin, jolloin verkon tilaukset ja julkaisut näkyvät niille kyseisen solmun tilauksina ja julkaisuina. Ne ovat siis domain-tilassa ja toimivat kohdan 8.4 mukaisesti. Katkoviivoitetut linjat kuvaavat olemassa olevia yhteyksiä, joita verkko ei käytä solmun 3 lähettämien viestien välitykseen.



Kuva 8.4: Verkossa esiintyvät yhteydet suhteessa solmun 3 käyttämiin yhteyksiin

Vastaavasti yhtenäiset viivat kuvaavat niitä yhteyksiä, joita verkko hyödyntää solmun 3 viestien välittämiseen. Valinnat käytettävistä yhteyksistä suorittaa reitityskerros, ja valintaperiaatteina ovat muun muassa yhteyden laatuun, ruuhkaisuuteen ja nopeuteen liittyvät seikat. Yksinomaan solmun läheisyys ei siis ratkaise käytettäviä yhteyksiä, eikä datapalvelimen yksikkö itse tee päätöstä käytettävistä reiteistä – ainoastaan kohteista, joille viesti välitetään.

Kun kuvan solmu 3 saa asiakasmoduuliltaan tilauksen lentotieto-nimisestä tiedosta, se lisää tilauksen itselleen. Tämän jälkeen se ilmoittaa uuden tilaustietonsa solmuille, joihin se on yhteydessä (2, 4 ja 7). Nämä solmut välittävät tilauksen ja tilaajan tiedon eteenpäin muille tuntemilleen solmuille. Myös julkaisutilanteessa toimitaan samalla tavalla.

Esimerkiksi kuvan mukaisessa tilanteessa solmu 4 välittää muuttuneet J&T-tietonsa solmuille 5, 7 ja 8. Tarkistamalla tietojen lähteen naapurisolmut voidaan havaita, ettei solmulle 7 tarvitse kertoa muutoksesta – lähde on jo kertonut saman muutoksen sille. Tämä ei poista kaikkea ylimääräistä liikennettä. Esimerkkinä turhasta liikenteestä olisi solmujen 8 ja 9 toisilleen välittämä, sama tieto muutoksesta solmun 3 tiedoissa. Naapureiden naapurien kyseleminen rasittaisi verkkoa kuitenkin liikaa, eikä niiden tietäminen auta tekemään luotettavia päätöksiä. Jos naapurin naapurit tiedettäisiin, solmu 8 voisi virheellisesti päätellä solmun 6 saavan tiedon muutoksesta solmun kaksi kautta. Tässä tilanteessa vain vastaanottajan katsotaan voivan luotettavasti havaita useaan kertaan saapuvat tiedot.

Varsinaisten tietojen lähetys eroaa tästä toimintamallista hiukan. Siinä jokaisen tietolähetyksen osalta kerrotaan kenelle kaikille se on menossa kyseistä reittiä pitkin. Tämä tieto pyydetään reitityskerrokselta, joka voi ilmoittaa kaikille kohteille parhaimmat reitit. Jos kuvan 8.4 esittämässä tilanteessa solmut 2, 6, 7 ja 9 tilaisivat lentotietoa solmulta 3, kertoisi reitityskerros solmun 6 olevan tavoitettavissa solmun 4 kautta ja solmun 9 vastaavasti solmun 7 kautta (sekä solmulle 2 olevan suoran yhteyden). Tällöin solmu 3 välittäisi tiedot taulukko 1:n mukaisesti. Taulukon sarake ”jäljellä” kuvaa jäljellä olevia kohteita, joille tieto on vielä menossa kyseistä reittiä ja tämä informaatio kulkee tietolähetyksen mukana. Nämä kohteet ovat datapalvelimen yksiköitä, eivätkä varsinaisia asiakkaita.

Taulukko 1: Reitityskerroksen tietojen pohjalta muodostetut viestit

kohde	jäljellä	data
solmu 2	2	lentotieto
solmu 4	6	lentotieto
solmu 7	7,9	lentotieto

Tämän informaation avulla jokainen tiedot vastaanottava solmu voi osana vastaanottoa tarkastaa pitääkö lähetys välittää vielä muille. Esimerkiksi solmu 4 pyytäisi tässä tilanteessa reitityskerrokselta parasta reittiä solmuun 6 ja välittäisi tämän jälkeen sen sitä pitkin eteenpäin. Solmu 7 toimisi samoin eli kysyisi reitin solmuun 9. Näin hyödynnetään uusinta tietoa ja varaudutaan muutoksiin verkon rakenteessa.

Tämä ei toisaalta optimoi verkon käyttöä – jos yhteys solmusta 2 solmuun 6 olisi kustannuksiltaan halvempi kuin solmun 3 käyttämä reitti solmujen 4 ja 8 kautta, tulisi solmun 6 tilauksen liittyä solmun 2 kanssa samaan ketjuun. Tämä tieto saataisiin selville pitämällä yllä kustannuskenttää tilauksen mukana. Tuota kustannuskenttää kasvatettaisiin jokaisella välillä, jossa ei jo välitetä tilattua tietoa. Tämän mahdollistamiseksi reitissä seuraavaan solmuun hyppäyksen mukanaan tuoma kustannus pitää saada reitityskerrokselta.

Näin solmun 6 tilauksen kustannukset reitille solmun 2 kautta olisi niiden välisen yhteyden kustannus (koska solmu 2 tilaa tietoa jo valmiiksi, ei hyppäyksestä solmujen 2 ja 3 välillä tule lisäkustannuksia). Vastaavasti kustannus reitille solmujen 4 ja 8 kautta sisältäisi kustannukset kaikista välin linkeistä, koska matkan varrella olevat solmut eivät vielä tilaa tietoa. Tämän kokonaiskustannuksen laskenta on helppo tehdä

datapalvelimen tasolla, sillä se hallinnoi tietovirtoja – näin verkkokerros ei joudu jatkuvasti laskemaan kustannuksia kaikkiin toisiin solmuihin. Vastaavasti yksittäisen linkin kustannuksen tietäminen on reitityskerroksen asia, sillä datapalvelin ei hallinnoi yksittäisiä linkkejä vaan reittejä.

Vaikka tämä voi johtaa solmun 6 näkökulmasta mahdollisesti hitaampaan toimintaan kuin käyttämällä solmujen 4 ja 8 kautta kulkevaa reittiä, on se todennäköisesti kokonaisuutena halvempaa. Vaihtoehtoisesti solmu 2 voi saada tiedon solmulta 6 pienemmällä kustannuksella kuin se saa sen suoraan solmulta 3. Tilanteen voi aiheuttaa esimerkiksi hidas ja epäluotettava radioyhteys solmujen 2 ja 3 välillä kun solmujen 2 ja 6 välillä on tehokkaammin ja luotettavammin toimiva linkki – tällöin solmun 2 tulisi saada tiedot solmun 6 kautta.

Koska solmun 2 tilaus on voinut olla voimassa ennen solmun 6 tilausta, sekä tilaukset että julkaisut kannattaa välillä päivittää. Näitä päivityksiä ei kannata tehdä kovin usein verkon ruuhkautumisen välttämiseksi. Toisaalta niitä ei ole hyvä tehdä harvoinkaan – tällöin verkon rakenne ja tilaustiedot voivat ehtiä muuttumaan turhan usein. Sopivan aikavälin valinta riippuu verkon koosta ja rakenteesta, eikä yksittäistä aina hyvää ratkaisua ole.

Verkon rakenteen muuttuessa voi myös tulla eteen tilanne, jossa reititys solmulle muuttuu kesken tiedon kulkemista reitillä. Tällöin se törmää jossain vaiheessa solmuun, johon ei löydy listan mukaista yhteyttä. Kysymällä reitityskerrokselta uutta reittiä solmun tavoittamiseksi vasta tässä vaiheessa, voidaan yhdistää molempien toimintatapojen hyvät puolet.

9. JOHTOPÄÄTÖKSET

Jako palvelimiin ja asiakkaisiin on ollut toimiva ratkaisu Internetin useissa käyttökohteissa. Kun asiakkaita on huomattavia määriä, on kuitenkin parempi käyttää vertaisverkkoteknologioita niiden paremman skaalautuvuuden vuoksi. Näin voidaan välttää kohdistamasta liian suurta rasitusta mahdollisesti ainoalle palvelimelle.

Vastaavasti julkaisija–tilaaja-mallin avulla voidaan paremmin hallita kenelle tietoja lähetetään. Nämä kaksi mallia on myös luontevaa yhdistää toisiinsa – jokainen vertaisverkossa oleva asiakas toimii julkaisijana hallussaan oleville tiedoille, ja tiedon tarvitsijat vastaavasti toimivat tilaajina kaikille julkaisijoille. Kertautuvia tietojen lähetyksiä voidaan pyrkiä vähentämään ketjuttamalla useiden asiakkaiden tilaukset yhdeksi tilausketjuksi.

Ei kuitenkaan ole yksioikoista yhdistää salattuja yhteyksiä tähän yhteyteen, koska salaukset estävät hyödyntämästä tietoa sen kulkeman matkan varrella. Käyttämällä salauksia eri hyppyjen välillä koko reitin sijasta voidaan ongelma kuitenkin välttää. Toisaalta tämä ei täytä vaativimpien tietoturvaluokkien tietojen vaatimuksia tiedon salauksien suhteen. Tällöin joudutaan luopumaan ketjutuksen tuomasta edusta.

Ehkä suurin haaste on ollut jako erilliseen reitityskerrokseen ja datapalvelimeen, koska datapalvelin joutuu osittain tekemään myös reititykseen liittyviä tehtäviä. Toisaalta tämä on kuitenkin myös yksinkertaistanut suunnittelua, koska tietojen välittämistapoihin ei ole tarvinnut kiinnittää huomiota. Jako erillisiin kerroksiin on kokonaisuutena hyödyllinen, joskin työnjakoon eri kerroksien välillä tulee kiinnittää paljon huomiota. Myös datapalvelimelle asetetut tietoturvavaatimukset ovat asettaneet omia haasteitaan suunnitelmalle.

Nämä haasteet ovat kuitenkin olleet ratkaistavissa, ja päivityssuunnitelma tietopalvelusta paremmin tulevaisuuden tarpeita vastaavaan datapalvelimeen on onnistuneesti tehty. Kohdatut haasteet ovat tehneet suunnitelman laadinnasta mielenkiintoa herättävän, sillä samalla on ollut mahdollista oppia lisää useista teknologioista.

Työssä esitetty suunnitelma päivityksestä tietopalvelusta datapalvelimeksi on pääosin pysynyt samana varsinaisen työajan päätyttyä, ja yrityksessä on toteutettu suunnitelmaan pohjautuva järjestelmän päivitys. Tämän jälkeen työstä on parannettu luettavuutta korjaamalla kielioppivirheitä sekä tuomalla paremmin esille tehtyjen valintojen taustalla olevat syyt. Päivitys datapalvelimeksi on tuonut mukanaan aiempaa tehokkaamman toimintatavan, ja siihen perustuva järjestelmä on osoittautunut tuotantokäytössä toimivaksi ratkaisuksi. Tämän vuoksi päivitykselle asetetut päämäärät voidaan katsoa tulleen täytetyiksi, vaikka kirjallinen osuus valmistui lopulliseen muotoonsa vasta varsinaisen päivityksen jo tapahduttua.

LÄHTEET

[1] **Salman A. Baset, Henning B. Schulzrinne:** "An Analysis of the Skype Peer-to-Peer Internet Telephony Protocol," [PDF], Computer Science Department of Columbia University, Technical report CUCS-039-04, 2004. 11 p. [viitattu 29.5.2009]. Saatavissa http://www1.cs.columbia.edu/~salman/publications/skype1_4.pdf

[2] **Alex Biryukov, Dmitry Khovratovich:** "Related-key Cryptanalysis of the Full AES-192 and AES-256," [PDF], University of Luxembourg, May 29th, 2009. 19 p. [viitattu 10.7.2009]. Saatavissa http://eprint.iacr.org/2009/317.pdf

[3] **Haowen Chan, Adrian Perrig, Dawn Song:** "Random Key Predistribution Schemes for Sensor Networks," [PDF], Computer Science school of Carnegie-Mellon University, CMU-CS-02-207, 21.4.2003. 28 p. [viitattu 4.6.2009]. Saatavissa http://reports-archive.adm.cs.cmu.edu/anon/2002/CMU-CS-02-207.pdf

[4] **Stephen J. Clayes:** "Announcing Request for Candidate Algorithm Nominations for a New Cryptographic Hash Algorithm (SHA-3) Family," [PDF], National Institute of Standards and Technology, docket No.: 070911510-7512-01, November 2nd, 2007. 9 p. [viitattu 10.7.2009]. Saatavissa http://csrc.nist.gov/groups/ST/hash/documents/FR_Notice_Nov07.pdf

[5] **Patrick TH. Eugster, Pascal A. Felber, Rachid Guerraoui, Anne-Marie Kermarrec:** "The Many Faces of Publish/Subscribe," [PDF], ACM Computing Surveys, Vol 35, No. 2, June 2003 pp. 114–131 [viitattu 30.9.2009]. Saatavissa http://www.irisa.fr/asap/intranet/the-many-faces-of-publish-subscribe.pdf/attachment_download/file

[6] **Federal Aviation Administration**: Chapter 7 of "Pilot's Handbook of Aeronautical Knowledge," [PDF], United States Department of Transportation, 2008 (updated September 2nd 2009). [Viitattu 16.4.2010]. Saatavissa http://www.faa.gov/library/manuals/aviation/pilot_handbook/media/PHAK - Chapter 07.pdf

[7] **Skand Singh Gupta, Shriram Rajagopalan:** "A measurement-based study of the Direct Connect Network," [PDF],Computer Science Department of California's, Santa Barbara University, 13.12.2007. 5 p. [viitattu 29.5.2009]. Saatavissa asdfhttp://www.cs.ucsb.edu/~shriram/DirectConnect-Report.pdf

[8] **Horst Feistel:** "Cryptography and Computer Privacy," [JPEG], Scientific American, Vol. 228, No. 5, May 1973 pp. 15–23. [viitattu 10.5.2010]. Saatavissa http://www.apprendre-en-ligne.net/crypto/bibliotheque/feistel/index.html

[9] **Ilkka Haikala, Hannu-Matti Järvinen**, 2004: "Käyttöjärjestelmät," Talentum, toinen painos, ISBN: 952–14–0851–0, 246 s.

[10] **Martin E. Hellman:** "An Overview of Public Key Cryptography," [PDF], IEEE Communications Magazine, 50th commemorative Issue, May 2002. Pages 42–49 [viitattu 24.6.2009]. Saatavissa http://www.cs.umanitoba.ca/~maheswar/anc2002/PAPERS/Hel78.pdf

[11] **Jason Lester Hill,** "System Architecture for Wireless Sensor Networks," [PDF], University of California, Berkeley, Spring 2003. 196 p. [viitattu 5.10.2009]. Saatavissa http://www.jlhlabs.com/jhill_cs/jhill_thesis.pdf

[12] **Stephen D. Huston, James CE Johnson, Umar Syyid**, February 2004: ”The ACE Programmer’s Guide: Practical Design Patterns for Network and Systems Programming,” Addison-Wesley, Pearson Education Inc., 2nd printing, ISBN: 0-201-69971-0, 506 p.

[13] **Panu A. Kalliokoski:** ”Säännöllisistä lausekkeista,” [PDF], Helsingin Yliopisto. 2004/02/05. 8 s. [viitattu 19.8.2009]. Saatavissa http://www.ling.helsinki.fi/kit/2004s/ctl130_kalliokoski/slausekkeet.pdf

[14] **Todd Lammle, Donald Porter & James Chellis**, 1999: “Cisco Certified Network Associate, Study Guide for exam 640-407,” Sybex Inc. ISBN: 0-7821-2381-3, 729 p.

[15] “MANET (Mobile Ad Hoc Network),” [WWW], Tech Terms Dictionary, 1 p. [viitattu 11.8.2009]. Saatavissa http://www.techterms.com/definition/manet

[16] **Benjamin McMillan:** “The BitTorrent Protocol,” [PDF], 10.3.2004. 3 p. [viitattu 1.6.2009]. Saatavissa http://www.lugatgt.org/articles/bittorrent/downloads/index.pdf

[17] **Johann van der Merwe, Dawoud Dawoud, Stephen McDonald:** ”Trustworthy Key Management for Mobile Ad Hoc Networks,” [PDF], Electrical, Electronic and Computer Engineering department of KwaZulu-Natal University, 2004. 6 p. [viitattu 2.6.2009]. Saatavissa http://www.ee.ukzn.ac.za/research/Pda Armscor Website/Pda Armscor Website/files/vanderMerwe-SATNAC04.pdf

[18] **D. Mitton, M. St.Johns, S. Barkley, D. Nelson, B. Patil, M. Stevens, B. Wolff:** “Authentication, Authorization and Accounting: Protocol Evaluation,” [WWW], Internet Engineering Task Force, RFC3127, June 2001. 88 p. [viitattu 3.6.2009]. Saatavissa http://www.ietf.org/rfc/rfc3127.txt?number=3127

[19] **Kaisa Nyberg:** “Kryptologia – tiedon turvaamisen tiede,” [PDF], Tietojenkäsittelytiede 26, Heinäkuu 2007 sivut 32–53 [viitattu 6.10.2009]. Saatavissa http://www.tkts.fi/lehti/a26/nyberg.pdf

[20] **Adrian Perrig, John Stankovic, David Wagner:** “Security in Wireless Sensor Networks,” [PDF], Communications of the ACM, Vol 47, No. 6, June 2004 pp. 53–58 [viitattu 5.10.2009]. Saatavissa http://www.cs.virginia.edu/papers/p53-perrig.pdf

[21] **B. Preneel, A. Biryukov, E. Oswald, B. van Rompay, L. Granboulan, E. Dottax, S. Murphy, A. Dent, J. White, M. Dichtl, S. Pyka, M. Schafheutle, P. Serf, E. Biham, E. Barkan, O. Dunkelman, J.-J. Quisquater, M. Ciet, F. Sica, L. Knudsen, M. Parker, H. Raddum:** “NESSIE Security Report, ” version 2.0, [PDF], EU Project ”Nessie” IST–1999–12324, document NES/DOC/ENS/WP5/D20/2, 19.2.2003. 342 p. [viitattu 4.6.2009]. Saatavissa https://www.cosic.esat.kuleuven.be/nessie/deliverables/D20-v2.pdf

[22] “Resource Access Control Facility Security Guide,” Version 3 Release 2, [PDF], IBM Corp., SC34-6835–00, 423 p. [viitattu 3.6.2009]. Saatavissa http://publib.boulder.ibm.com/infocenter/cicsts/v3r2/topic/com.ibm.cics.ts.doc/pdf/dfht5c00.pdf

[23] **Henry S. Thompson, David Beech, Murray Maloney, Noah Mendelsohn:** ”XML Schema Part 1: Structures Second Edition,” [WWW] W3C Recommendation, 28.10.2004. 56 p. [viitattu 2.6.2009]. Saatavissa http://www.w3.org/TR/2004/REC-xmlschema-1-20041028/

[24] ”Tietoaineistoturvallisuus valtionhallinnossa,” [PDF], Valtiovarainministeriö, Vahti-luonnos, v1.0, 11.12.2008. 73 s. [viitattu 1.6.2009]. Saatavissa http://www.vm.fi/vm/fi/04_julkaisut_ja_asiakirjat/03_muut_asiakirjat/20070717Lausun/TietoaineistoturvallisuusValtionhallinnossa-ver_11122008.pdf

[25] ”Valtionhallinnon keskeisten tietojärjestelmien turvaaminen,”, [PDF], Valtiovarainministeriö, Vahti 5/2004, 13.12.2004. 112 s. [viitattu 2.6.2009]. Saatavissa http://www.vm.fi/vm/fi/04_julkaisut_ja_asiakirjat/01_julkaisut/05_valtionhallinnon_tietoturvallisuus/90727_fi.pdf

[26] **Antti Viidanoja**, 2009**:** ”Mesh-verkon reitittimen toteutus,” julkaisematon diplomityön esiversio, Tampereen Teknillinen Yliopisto, Tietotekniikan koulutusohjelma.

[27] **Juliette White:** “Initial report on the LEVIATHAN stream cipher,” [PDF], NESSIE, document NES/DOC/RHU/WP3/009/2, 2.2.2001. 4 p. [viitattu 5.6.2009]. Saatavissa https://www.cosic.esat.kuleuven.be/nessie/reports/phase1/rhuwp3-009-2.pdf

[28] **Yihui Zhang, Li Xu, Xiaoding Wang:** “A Cooperative Secure Routing Protocol based on Reputation System for Ad Hoc Networks,” [PDF], Journal of Communications volume 3 number 6 (pages 43–50), November 2008. [viitattu 9.6.2009]. Saatavissa http://www.academypublisher.com/jcm/vol03/no06/jcm03064350.pdf

LIITE 1: TILAUKSEN ETENEMINEN UML-SEKVENSSINÄ